Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

перший (бакалаврський)
(рівень вищої освіти)

Розробка системи автоматизації для ідентифікації виробів за допомогою машинного зору
(тема)

Виконав:
студент 4 курсу, групи АКТАКІТ-20-2
Гасанов Р. М.
(прізвище, ініціали)
Спеціальності 151 Автоматизація та комп`ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Автоматизація та комп`ютерно-інтегровані технології
(повна назва освітньої програми)
Керівник доц. Аллахверанов Р. Ю.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри КІТАР _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2024р.

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
Рівень вищої освіти перший (бакалаврський)
Спеціальність 151 Автоматизація та комп`ютерно-інтегровані технології
Тип програми освітньо-професійна
Освітня програма Автоматизація та комп`ютерно-інтегровані технології
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри КІТАР _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові *Гасанову Рустаму Мірдаметовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Розробка системи автоматизації для ідентифікації виробів за допомогою машинного зору*
затверджена наказом по університету від " 03" червня 2024 р. №544 Ст.
2. Термін подання студентом роботи " 27 " червня 2024р.
3. Вихідні дані до роботи *3.1 Промисловий робот з системою технічного зору;*
3.2 Бібліотека Integrated Performance Primitives (IPP);
3.3 Бібліотека AviCap;
3.4 Бібліотека OpenCV;
3.4 Оформлення текстової документації згідно ДСТУ 3008-2015.

4. Перелік питань, що потрібно опрацювати в роботі *4.1 Вступ;*
4.2 Методи обробки інформації у системах технічного зору роботів;
4.3 Методи ідентифікації виробів у роботизованих системах;
4.4 Практична реалізація методів обробки інформації у системах технічного зору;
4.5 Розробка програмного забезпечення для визначення параметрів виробів поточного виробництва за допомогою машинного зору;
4.6 Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій Демонстраційний матеріал представлений у форматі презентації PowerPoint (*.ppt) – *16 с. формату А4*

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	*Методи обробки інформації у системах технічного зору роботів*	*13.05 – 18.05.24*	*виконано*
2	*Методи ідентифікації виробів у роботизованих система*	*19.05 – 24.05.24*	*виконано*
3	*Практична реалізація методів обробки інформації у системах технічного зору*	*25.05 – 30.05.24*	*виконано*
4	*Розробка програмного забезпечення для визначення параметрів виробів поточного виробництва за допомогою машинного зору*	*31.05 – 09.06.24*	*виконано*
5	*Промислова безпека та аналіз умов праці на робочому місці*	*10.06 – 12.06.24*	*виконано*
6	*Оформлення пояснювальної записки*	*13.06 – 15.06.24*	*виконано*
7	*Подання роботи на перевірку Інтернет-системою StrikePlagiarism*	*16.06 – 19.06.24*	*виконано*
8	*Подання роботи на рецензію*	*20.06 – 22.06.24*	*виконано*
9	*Подання роботи на підпис зав. кафедри*	*23.06 – 25.06.24*	*виконано*
10	*Подання кваліфікаційної роботи в ЕК*	*26.06.24*	*виконано*

Дата видачі завдання 13 травня 2024 р.

Студент _____ Гасанов Р. М.
(підпис)

Керівник роботи _____ доц. Аллахверанов Р. Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 104 с., 8 табл., 10 рис., 3 дод., 15 джерел.

ПРОМИСЛОВИЙ РОБОТ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, СИСТЕМА ТЕХНІЧНОГО ЗОРУ, РОЗПІЗНАВАННЯ, ІДЕНТИФІКАЦІЯ, МЕТОДИ ОБРОБКИ ІНФОРМАЦІЇ, ПОТОЧНЕ ВИРОБНИЦТВО.

Об'єкт дослідження – розпізнавання та ідентифікації виробів поточного виробництва, за допомогою машинного зору промислового робота.

Предмет дослідження – програмне забезпечення.

Мета роботи – розроблення програмного забезпечення для системи розпізнавання та ідентифікації виробів, за допомогою машинного зору промислового робота.

Розглянуто методи розпізнавання та ідентифікації виробів поточного виробництва, проаналізовано методи обробки інформації у системах машинного зору роботів, побудована теоретико-множинна модель розпізнавання та ідентифікації, розглянута практична реалізація методів обробки інформації в робототехнічних системах, розроблено програмне забезпечення, що реалізує методи розпізнавання та ідентифікації виробів поточного виробництва.

До числа перспективних напрямків досліджень слід віднести подальший розвиток методів розпізнавання та ідентифікації, їх програмну реалізацію для промислових операційних систем реального часу. У комбінації з системою прийняття рішень такі розробки істотно прискорять реалізацію систем управління інтелектуальними роботами.

ABSTRACT

Explanatory note: 104 pp., 8 tab., 10 figs., 3 appendices, 15 sources.

INDUSTRIAL ROBOT, SOFTWARE, TECHNICAL VISION SYSTEM, RECOGNITION, IDENTIFICATION, INFORMATION PROCESSING METHODS, CURRENT PRODUCTION.

The object of the research is the recognition and identification of products of current production, using the machine vision of an industrial robot.

The subject of research is software.

The purpose of the work is to develop software for the system of recognition and identification of products, using the machine vision of an industrial robot.

The methods of recognition and identification of details of current production were considered, the methods of information processing in systems of technical vision of robots were analyzed, a theoretical-multiple recognition and identification model was built, the practical implementation of information processing methods in robotic systems was considered, software was developed that implements methods of recognition and identification of details of current production.

Further development of recognition and identification methods, their software implementation for real-time industrial operating systems should be included among the promising areas of research. In combination with a decision-making system, such developments will significantly accelerate the implementation of intelligent robot management systems.

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«20» червня 2024 р.  Гасанов Р. М.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ЕОМ – електронно-обчислювальна машина;

КПО – коефіцієнт природної освітленості;

ПСО – перетворення середніх осей;

РС – роботизована система;

МЗ – система технічного зору;

ШНМ – штучні нейронні мережі;

Avi Cap – avi capture;

IPP – integrated performance primitives (бібліотека комп'ютерного зору);

MLL – machine learning library;

Open CV – open source computer vision library (бібліотека комп'ютерного зору з відкритим вихідним кодом).

ВСТУП

Машинний зір (МЗ) робота, здебільшого, визначають як процес виділення й ідентифікації, а також перетворення інформації, отриманої від тривимірних зображень. Такий процес має й іншу назву – технічний зір. Складається він з шести ключових етапів:

– зчитування інформації;

– попередньої обробки інформації;

– сегментації;

– опису;

– розпізнавання;

– інтерпретації.

Процес зчитування інформації реалізується шляхом отримання візуального зображення. Попередню обробку інформації доцільно виконувати на базі методів, зокрема, поліпшення зображення окремих виробів чи зниження шуму. Під поняттям «сегментація» мається на увазі процес позначення виробів, які викликають зацікавлення, на зображенні. Виконуючи опис, формулюються параметри, що характерні для виділення необхідного об’єкта, зокрема розмір або форма, що відрізняють його від інших. Процес розпізнавання базується на ідентифікації виробів (блоку двигуна, болтів, гайкового ключа). Під інтерпретацією мається на увазі приналежність розпізнавальних виробів до групи.

Зазначені вище етапи доцільно об’єднати, беручи до уваги складність відповідно до їхньої реалізації. З огляду на це, можна виокремити МЗ трьох рівнів: низький, середній та високий. Зауважимо, що між названими рівнями не існує чітких кордонів. Однак, оскільки все одно різні процеси систем підлягають класифікації, то їхнє виокремлення має підґрунтя.

Процеси МЗ низького рівня, з точки зору реалізації автоматичних дій, доволі прості, тому штучний інтелект до них не застосовують. Зчитування та попередня обробка інформації теж уважаються процесами даного рівня.

Середній рівень МЗ характеризується такими процесами, як виділення, ідентифікація, а також розмітка елементів зображення, що надійшли з нижнього рівня. Ґрунтуючись на зазначеному вище розподілі МЗ, то опис та розпізнавання окремих виробів, сегментація теж відповідають середньому рівневі.

Процеси, що взаємодіють зі штучним інтелектом, доцільно віднести до високого рівня МЗ.

Об'єкт дослідження – розпізнавання та ідентифікації виробів поточного виробництва, за допомогою машинного зору промислового робота.

Предмет дослідження – програмне забезпечення.

Мета роботи – розроблення програмного забезпечення для системи розпізнавання та ідентифікації виробів, за допомогою машинного зору промислового робота.

Для досягнення поставленої мети виконаємо такі завдання, як:

– проаналізуємо основні методи розпізнавання й ідентифікації;

– побудуємо теоретико-множинну модель процесу розпізнавання й ідентифікації;

– розробимо програмне забезпечення, що ґрунтується на розпізнаванні й ідентифікації простих виробів.

Пояснювальну записку з кваліфікаційної роботи оформлено згідно з ДСТУ 3008:2015 [1], а також з рекомендаціями з підготовки і оформлення кваліфікаційної роботи здобувачами першого (бакалаврського) рівня вищої освіти [2-3].

1 МЕТОДИ ОБРОБКИ ІНФОРМАЦІЇ У СИСТЕМАХ МАШИННОГО ЗОРУ РОБОТІВ

1.1 Класифікація методів обробки інформації у системах машинного зору

Робототехнічні системи завжди характеризуються одним із найважливіших напрямків розвитку, зокрема, організацією взаємодії маніпулятора та сенсорними аналізаторами. Наявність сенсорів сприяє гнучкості системи, розширенню її функціональних можливостей, а також збільшується коло завдань, які вона розв’язує. Як правило, переважна більшість з систем, які функціонують на взаємодії маніпулятора з сенсорами, розроблені на основі зв'язування автономних сенсорної підсистеми та робототехнічної. Зазначені системи демонструють втілення концепції взаємодії системи керування роботом із сенсорною підсистемою, що є основою для розробленої концепції візуального керування роботами [5-6]. Схему обробки інформації в МЗ робота продемонстровано на рисунку 1.1.

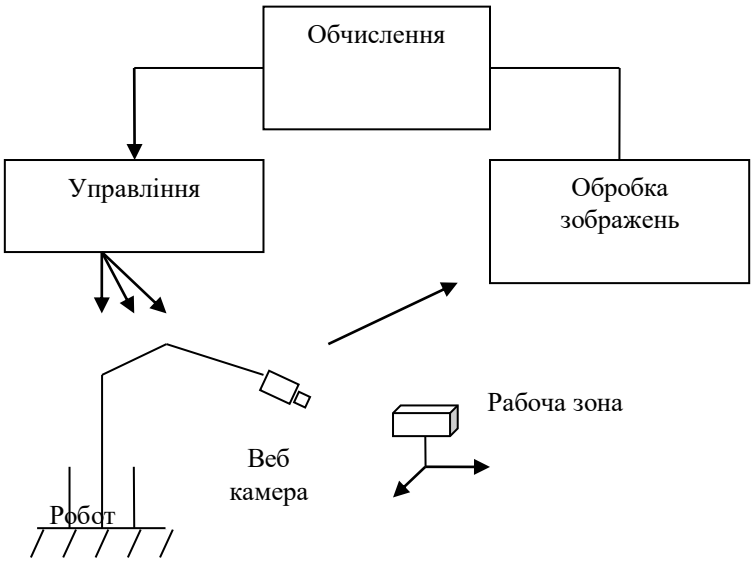

Рисунок 1.1 – Обробка інформації в МЗ робота

Для розмежування методів і підходів, наявних у системах МЗ, виникає потреба розподілити зір за трьома основними підкласами: низьким, середнім та високим рівнями. На низькому рівні системи машинного зору відповідають за обробку інформації, що надходить з датчиків зчутливлювання [6].

Зазначені системи належать до класу «інтелектуальних» машин за умови наявності таких характеристик, як:

– можливість видокремлення суттєвої інформації з безлічі незалежних ознак;

– здатність навчатися на прикладах й узагальнення цих знань з метою подальшого їхнього застосування в нових ситуаціях;

– можливість щодо відновлення подій за умови опрацювання неповної інформації;

– здатність визначати цілі та формулювати плани щодо їхнього досягнення.

Побудова систем МЗ, що містять наробки для обмежених видів робочого простору, теоретично є можливою, але особливості таких систем відрізняються від реальних можливостей людського зору. Основа машинного зору базується на аналітичній формалізації. Вона спрямована на розв’язання конкретних завдань. Проте машини з сенсорними характеристиками, що аналогічні можливостям людини, скоріш за все, з’являться не скоро. Звернемо увагу, що копіювання природи не єдиний варіант для розв’язання зазначеної проблеми [6]. У сучасному розумінні задачі польоту у просторі мають зовсім інше розв'язання, на відміну від тих, які підказані природою. З огляду на це, літаки значно перевершують можливості птахів щодо швидкості та набору висоти.

Системи МЗ середнього рівня виконують завдання щодо сегментації, опису та розпізнавання окремих об'єктів. Як правили, на цьому рівні задіяно безліч підходів, які ґрунтуються на аналітичних поданнях. Системи МЗ високого рівня розв’язують проблеми, що вже окреслювались вище. З метою

усвідомлення проблем МЗ високого рівня, а разом з тим і його зв'язку як з низьким, так і середнім рівнями, здебільшого вводять низку обмежень для спрощення розв'язуваних задач.

1.2 Методи попередньої обробки зображень

Для ідентифікації об'єктів, які наявні в робочій зоні робота, зазвичай виконується два етапи: по-перше, необхідно сформулювати ознаки, котрими наділені об'єкти; по-друге, здійснити розпізнавання об'єктів, ґрунтуючись на виявленій сукупності характерних ознак. Якщо дотримуватись згаданої структури, процес з ідентифікації алгоритму обробки інформації в МЗ доцільно розподілити на алгоритми попередньої обробки, а також алгоритми розпізнавання. Певною мірою такий розподіл можна назвати умовним, тому що застосуваннях одних і тих же алгоритмів за математичним призначенням на практиці можуть бути використані на обох етапах зазначеного процесу.

З огляду на спектральний діапазон перетворювачів світло-сигнал та принцип дії МЗ формування зображення робочої зони робота виконується датчиком. У даному випадку зображенням є двовимірна картина поля інтенсивності випромінювань робочої зони. Формально отримуємо зображення таким чином: визначаємо функціональні залежності інтенсивності випромінювань робочої зони від координат точок зображення x і y:

$$J = f(x, y). \tag{1.1}$$

У подальшому визначатимемо зображення як функціональну залежність $f(x, y)$. Спираючись на опрацьовану термінологію, окреслимо завдання попередньої обробки як пошук будь-яких особливостей функції $f(x, y)$, які вказуватимуть на тип об'єкта в межах робочої зони [6].

1.3 Сегментація виробів

1.3.1 Проведення контурів і визначення меж

Сегментацією називається поділ зображення на об’єкти чи складові частини. Крім того, сегментацію визначають як один із ключових елементів роботи автоматизованої системи МЗ, бо якраз на цьому етапі обробки на зображенні необхідно виділити об'єкти, щоб далі їх проаналізувати та розпізнати. Алгоритми сегментації, здебільшого, засновані на двох ключових принципах: розривність та подоба. За основу розривності береться визначення контурів, а подоба ґрунтується на встановленні порогового рівня та розширенні області. Такі поняття характерні статичним і динамічним зображенням. Зауважимо, що для поліпшення роботи алгоритмів сегментації рух – потужний засіб.

Обчислення градієнта, граничний поділ, який визначає розриви в інтенсивності подання образу об'єкта – все це називається методами. Пріоритетно, якщо зазначені методи визначають пікселі, що розташовані на межі фону й об’єкта. Проте такі пікселі зрідка повністю характеризують межі через шум, розриви на межі через нерівномірну освітленість чи інші ефекти, що призводять до розмиття зображення на практиці. Таким чином, алгоритми з визначення контурів реалізуються процедурами побудови меж об'єктів відповідною послідовністю пікселів [7]. Методики, що допоможуть досягти заданої мети, розглянемо нижче.

1.3.2 Локальний аналіз

Одним із елементарних підходів з'єднання точок контуру вважаємо аналіз характеристик пікселів у невеликому околі (наприклад, в околі розміром 3×3 або 5×5) кожної точки (x, y) образу, в якому попередньо вже виявлено контур. Всі подібні точки з'єднані, таким чином, утворено межу з пікселів, які характеризуються деякими загальними властивостями.

Для проведення аналізу щодо встановлення подібності пікселів контуру необхідно обчислити:

– величину градієнта для побудови контурного пікселя;

– напрямок градієнта.

Позначимо першу характеристику величиною $G\{f(x, y)$:

$$G[f(x,y)] = [G^2{}_x + G_y^2]^{1/2} = [(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial x})^2]^{1/2},$$
$$G[f(x,y)] \cong |G_x| + |G_y|. \tag{1.2}$$

Піксель контуру з координатами (x', y') є подібним за величиною в окресленому раніше околі (x, y) пікселя з координатами (x, y), якщо нерівність виявляється справедливою

$$|G[f(x,y)] - G[f(x',y')]| \le T, \tag{1.3}$$

де T – порогове значення.

Напрямок градієнта обчислимо за кутом вектора градієнта, що отримали із рівняння

$$G[f(x,y)] = \begin{bmatrix} G \\ G_y \end{bmatrix}, \tag{1.4}$$

$$\theta = arctg\begin{bmatrix} G_x \\ G_y \end{bmatrix}. \tag{1.5}$$

де θ – кут (щодо осі x), уздовж якого найбільше значення належить швидкості зміни.

У результаті кут пікселя контуру з координатами $\{x', y'\}$ в деякому околі (x, y) є подібним до кута, в якому пікселі з координатами $\{x, y\}$, якщо дотримуватись такої нерівності:

$$|\theta - \theta'| < A, \tag{1.6}$$

де A – порогове значення кута.

Звертаємо увагу, що напрямок контуру в точці $\{x, y\}$ насправді є перпендикулярним напрямку вектора градієнта в цій точці. Хоча для порівняння напрямків нерівність подає еквівалентні результати.

Ґрунтуючись на даних припущеннях, виконуємо з'єднання точки в деякому околі (x, y) з пікселем, координати котрого позначаються як $\{x, y\}$, якщо критерії задовольняються величиною та напрямком. Під час руху від пікселя до пікселя та за умови врахування, що кожна точка приєднується як центр околу, процес виконується для кожної точки образу. У зв’язку з цим залучається стандартна бібліотечна процедура, в ході якої між рівнями інтенсивності освітлення та послідовностями пікселів контуру встановлюється відповідність.

Для створення якісного зображення, перш за все, необхідно з’ясувати розміри прямокутників, чітко визначивши горизонтальні та вертикальні контури. У подальшому виконується поєднання сегментів контуру (вони розділені невеликими проміжками), після чого необхідно об'єднати окремі короткі сегменти.

1.3.3 Глобальний аналіз із використанням перетворення Хоуга

Метод з'єднання граничних точок пояснюється шляхом визначення їхнього розміщення на кривій спеціального виду. Першочергово припустимо, що на площині xy образу дано n точок, отже, визначимо підпослідовності точок на прямих лініях. Як одне із можливих рішень розглянемо процес побудови всіх ліній, які проходять через кожну пару точок. Далі знайдемо всі

підпослідовності точок, наближених до певних ліній. Таким чином, маємо завдання, пов'язані з цією процедурою, спочатку з'ясуємо розташування $n(n-1)/2 \sim n^2$ ліній, далі виконаємо $n[n(n-1)]/2 \sim n^3$ порівняння кожної точки з усіма лініями. Зауважимо, що даний процес (з обчислювальної точки зору) доволі трудомісткий за винятком найпростіших застосунків [7].

Проте для розв'язання задачі можемо застосувати інший підхід, котрий запропонував Хоуг. Іншими словами, мається на увазі перетворення Хоуга. Перевагою перетворення Хоуга в обчислюванні вважатимемо поділ простору параметрів на так звані збираючі елементи. У них ($a_{макс}$, $a_{мін}$) та ($b_{макс}$, $b_{мін}$) – це допустимі величини параметрів ліній. Збираючий елемент A (i, j) подається як площа, котра пов'язується з координатами простору параметрів (a_i, b_j). На початковому етапі ці елементи розглядаються як рівні нулю. З огляду на це для кожної точки (x_k, y_k) *в* у площині образу доцільно припустити, що параметр a дорівнює кожному з допустимих значень на осі a. Відповідне b обчислюємо, шляхом використання рівняння $b = -x_k + y_k$. За результатами виконаних дій отримане значення b округляється до найближчого допустимого значення на осі b. За умови, що вибір a_p потребує обчислення b_q, відповідно, розрахуємо $A(p, q)=A(p, q) + 1$. За результатами даної процедури в елементі A (i, j) значення M відповідає точкам у площині xy, розміщеним на лінії $y=a_ix+b$. Констатуємо, що точність знаходження точок на одній прямій залежить від числа розбиття площини ab. У разі розібиття вісі а на K частин для кожної точки (x_k, y_k) буде існувати K значень b, відповідних K можливих значень а. Оскільки точок образу налічується n, відповідно, у процесі має відбутись nK обчислювальних операцій. Розглянута процедура є лінійною до n ,до того ж, з меншою кількістю обчислювальних операцій, за умови, що $K <= n$.

1.4 Визначення порогового рівня зображень

Поняття порогового рівня (порога) T представлено як

$$T = T\,[x,\ y,\ p\ (x,\ y),\ f\ (x,\ y)], \tag{1.7}$$

де $f(x,\ y)$ окреслює інтенсивність у точці $(x,\ y)$;

$p(x,\ y$ позначає деяку локальну властивість у межах околі цієї точки.

Граничне зображення представимо як:

$$g(x,y) = \begin{cases} f(x,y) > T \\ f(x,y) \le T \end{cases}, \tag{1.8}$$

в якому пікселі в g(x, y) зі значенням 1 відповідатимуть об'єктам, а пікселі зі значенням 0 – фону. Передбачено у рівнянні, що в об'єктів інтенсивність більша за інтенсивність фону. Реалізація протилежної умови полягає у зміни знаків у нерівностях.

1.5 Обласна-орієнтована сегментація

1.5.1 Основні номінації

Мета сегментації визначається шляхом розподілу образу на області. Проаналізуємо методи сегментації, спрямовані на пряме знаходження областей [7].

Представимо R як область образу. У такому разі сегментація єпроцесом розбиття R на n підобластей $R_1,\ R_2,\ ...,\ R_n,$ отже:

$$\bigcup_{i=1}^{n} Ri = R\,,$$

де P_i – зв'язна область, i= 1, 2, ..., n;

$R_i = \varnothing$ для всіх і та j, $i \neq j$;

$P(R_i)$ – ІСТИНА для i= 1, 2, ..., n;

$P(R_iUR_i)$ – ХИБНІСТЬ для $i \neq j$, де $P(R_i)$ подано як логічний предикат, який визначено на точках із множини R_i, i $\varnothing$ – порожня множина.

Умова 1 зобов'язує до повної сегментації. Мається на увазі, що кожен піксель повинен знаходитись в образі. Умова 2 зобов'язує, щоб в області точки були зв'язковими. Умова 3 вимагає, недопустимість ймовірності перетину областей. Умова 4 окреслює властивості, котрі повинні задовольняти пікселі в сегментованій області. Наведемо приклад: $P(R_i)$ = ІСТИНА, якщо всі пікселі в R_i мають однакову інтенсивність. Умова 5 визначає розрізнення областей R_i та R_i за предикатом P.

1.5.2 Розширення області допомогою об'єднання пікселів

Розширення області виконують за допомогою групування пікселів або підобластей у великі об'єднання. Найпростіше – це агрегування пікселів. Для початку необхідно вибрати багато вузлових точок, за їхньою допомогою слід виконати розширення області, приєднавши до них сусідні пікселі зі схожими характеристиками, зокрема, текстура, колір, інтенсивність. Зауважимо, що може виникнути дві проблеми. Одна пов’язана із вибором початкових вузлів для правильного уявлення областей, що викликають зацікавлення, а друга - із визначенням необхідних властивостей для включення точок до різних областей під час розширення. Вибрати множину з однією чи кількома початковими точками залежить від того, як поставлено задачу. На кшталт, у військових програмах об'єкти, до яких виникає зацікавлення, мають значно вищу температуру за фон, тому виділяються більш яскраво. Вибір найяскравіших пікселів уважається початковим кроком для алгоритму розширення області. У разі відсутності апріорної інформації можна розпочинати обчислення набору властивостей для кожного пікселя. Цей набір, вірогідно, буде використано з метою встановлення відповідності пікселя тій чи іншій області під час розширення. Якщо результат обчислень

окреслюватиметься групами точок (кластерами), відповідно, вузловими вважатимуться пікселі з наближеними властивостями до властивостей центроїдів цих груп. Гістограма інтенсивностей, наприклад, показала б, що точки, котрі містять інтенсивність від одного до семи, набувають рис домінуючих. Власне вибору критерію подібності сприяє вид даних щодо образу, а також завдання. Зауважимо, що на аналіз інформації, що надходить зі супутників, істотно впливає колір, тому під час використання тільки монохроматичних образів, завдання аналізу значно ускладнюватиметься. На сьогодні для промислового МЗ скоріше виняток, ніж правило є можливість отримання мультиспектральних та інших додаткових даних про спосіб [7]. Здебільшого аналіз області виконується за допомогою набору дескрипторів, яким характерні інтенсивність і просторові особливості (моменти, текстури) одного джерела зображення. Важливо, що застосування лише одних дескрипторів може сприяти отриманню неправильних результатів, якщо нехтувати інформацією про умови зв'язку під час розширення області. Це легко продемонструвати. Наведемо приклад випадкового розташування пікселів із трьома різними значеннями інтенсивності. У разі об'єднання пікселів в «область» за показниками однакової інтенсивності, не звертаючи увагу на умови зв'язку, під час сегментації отримаємо безглуздий результат.

Крім того, під час розширення області виникає наступна проблема, котру неможна оминути, є формулювання умови закінчення процесу. Здебільшого закінчення процесу розширення області виникає за умови відсутності пікселів, які відповідають критерію приналежності до однієї чи іншої області. Раніше ми вже розглянули такі критерії, як текстура, колір, інтенсивність та з’ясували, що за своєю природою вони локальні. Разом з тим, не враховують «історію» процесу розширення області. У свою чергу, додатковий критерій щодо можливостей підвищення потужності алгоритму розширення області сформований на таких поняттях, як розмір, схожість між пікселем-кандидатом і щойно створеними пікселями (порівняння інтенсивності кандидата та середньої інтенсивності області), форма області,

що потребує розширення. З огляду на це, такі типи дескриптора застосовуються, ґрунтуючись на припущенні щодо неповної інформації про очікувані результати.

1.5.3 Розбиття та об'єднання області

Робота процедури розширення області розпочинається з заданої множини вузлових точок. Хоча першим етапом можна вважати розбиття образу на низку довільних непересічних областей, а далі вже об'єднувати та/або розбивати ці області доти, поки не будуть задовільнені умови. Ітеративні алгоритми розбиття й об'єднання, можуть реалізовуватись таким чином, що спрямовуються на виконання цих обмежень [6]. На кожному кроці необхідно виконувати такі операції:

– розбити область *Ri* на чотири непересічних квадранта, де *P (Ri)* = ХИБНІСТЬ;

– об'єднати сусідні області *Ri* і *Rk,* де *P (Ri U Rk)* = ІСТИНА;

– призупинити процес, якщо неможливе подальше розбиття чи об’єднання.

1.5.4 Аналіз руху об'єктів

Рух можна визначити як потужний засіб, який використовує тварина чи людина. За допомогою руху зручно виокремлювати об’єкти на фоні. У системах МЗ (машинного зору) роботів рух є складовим елементом під час активізації різних операцій на конвеєрі, у процесі переміщення руки, обладнаної датчиком, рідше – під час пересування всієї робототехнічної системи.

Серед найпростіших підходів, що виявляють зміни між двома кадрами зображення (образами) $f(x, y, t_i)$ і $f(x, y, t_j)$, та взятими, відповідно, в період часу t_i и t_j, можемо говорити про порівняння відповідних пікселів цих двох

образів. У даному випадку реалізується процедура з формування так званої різниці образів.

Розглянемо, наприклад, еталонний образ, в якому присутні лише стаціонарні компоненти. Якщо цей образ порівняти з таким, в якому наявні рухомі об'єкти, то різницею двох образів як результат, буде викреслювання стаціонарних компонент. Іншими словами, залишаться лише ненульові записи, котрі відповідають нестаціонарним компонентам зображення) [7].

1.6 Проблема опису об'єктів

Проблема опису у МЗ трактується як виділення властивостей (виробів) об'єкта для процесу розпізнавання. За ідеальних умов дескриптори не повинні мати залежність від розташування, розмірів та орієнтації об'єкта. З іншого боку повинні володіти інформацією в необхідній кількості з метою надійної ідентифікації об'єктів. Під час конструювання систем МЗ опис – це головний результат. У зв’язку з цим дескриптори матимуть вплив не тільки на складність алгоритмів розпізнавання, проте й на їхню роботу. Отже, доцільно виокремити три основні категорії дескрипторів: дескриптори межі, області, а також дескриптори для опису тривимірних структур.

1.7 Дескриптори межі

У ланцюгових кодах подання межі застосовують у вигляді послідовності відрізків прямих ліній, в яких має бути певна довжина та необхідний напрямок. Здебільшого, обирають 4 або 8-зв'язкову прямокутну решітку, що є основою цього уявлення. Довжину кожного відрізка обчислюють роздільною здатністю решітки, а напрямки – обраним кодом. Звертаємо увагу, що для реалізації всіх напрямків у 4-направленому ланцюговому коді достатньо лише 2 біт, для 8-спрямованого ланцюгового – 3. Створення ланцюгового коду заданої межі розпочинається з

першочергового вибору решітки. Площа комірки, розташованої усередині межі, може бути більшою за визначене число (як правило, 50 %), тому її наділяють значенням 1, у разі противаги – значенням 0. Наступний етап полягає в кодуванні межі між двома областями із залученням напрямків. Результат кодування закладено в напрямку за годинниковою стрілкою. Його початок знаходиться у місці, що позначено точкою. Крім того, існує процедура щодо розбиття межі на ділянки з рівною довжиною (однаковим числом пікселів наділяється кожна ділянка). Разом з тим мова йде і про з'єднання прямою лінією граничних точок кожної ділянки. В результаті кожній лінії необхідно з’ясувати напрямок. Як правило, ним стає найближчий до одного з допустимих напрямків ланцюгового коду. До того ж, ланцюговий код межі має пряму залежність від початкової точки. Хоча є можливість нормувати його елементарною процедурою. Початкова точка на решітці для створення ланцюгового коду вибирається у довільному порядку. Оскільки ланцюговий код ми розглядаємо з точки зору замкнутої послідовності індексів напрямків, то початкову точку необхідно вибирати з урахуванням того, що вислідна послідовність індексів – ціле число з мінімальною величиною. Зауважимо, якщо замість ланцюгового коду застосовувати його першу різницю, то виникає можливість нормування поворотів. Обчислення виконується шляхом відліку кількості напрямків (проти годинникової стрілки), що розмежовують два сусідніх елементи коду. Нормування можна реалізовувати таким чином: спочатку розбити всі кордони об'єкта на однакову кількість рівних відрізків, після чого підігнати довжини сегментів коду (вони повинні відповідати цьому розбиванню).

Сигнатурою називаємо одномірне функціональне уявлення межі [8]. Однак способів щодо створення сигнатур існує декілька. Найелементарніший – це побудова відрізка від центру до межі, тобто функції кута. Здебільшого побудова сигнатури базується на периметрі області та початкової точки. Виконати нормування периметра таким чином: пронормувати криву $r(\theta)$ максимальним значенням. Постає питання вибору

початкової точки. Першим кроком є визначення ланцюгового коду межі. Другий крок – застосувати метод із попереднього розділу. Використовувати відстань, яка залежить від кута, для визначення сигнатури як єдиний спосіб – недоцільно. Як варіант, розглянемо проведення прямої лінії через межу. Це дозволяє визначити кут між дотичною до межі та лінією як функцію положення вздовж меж. Така сигнатура різнитиметься з кривою $r(\theta)$, але буде наділена інформацією про ключові характеристики форми межі. Уявімо, горизонтальні ділянки кривої відповідають прямим лініям вздовж меж, бо незмінним є кут дотичної. В одному із варіантів цього методу під сигнатурою застосовується функція щільності нахилу, котра подається як гістограма значень кута дотичної. З огляду на те, що гістограма окреслюється як міра концентрації величин, функція щільності нахилу точно відповідає ділянкам межі з постійними кутами дотичній. Це прямі або майже прямі ділянки, іноді виникають глибокі провали для ділянок, які відповідають за швидкі зміні кутів (виступи або інші види вигинів).

Задачі апроксимації багатокутниками залучають методи об'єднання, котрі реалізуються на помилці чи інших умовах. Так, одним із підходів є з'єднання точок межі лінією за методом найменших квадратів. Лінію слід продовжувати, доки помилка апроксимації не перевищить раніше заданий поріг. У разі перевищення порогу, параметри лінії занотовуються у пам'яті, помилка дорівнює нулю, а процедура повторюється. З’єднання нових точок триває до тих пір, поки помилка знову не перевищить поріг. Кінцевим етапом процедури є утворення вершини багатокутника в результаті перетину сусідніх ліній. Проте такий підхід створює значне ускладнення, у зв’язку з тим, що вершини, здебільшого, не відповідають вигинам межі (на кшталт кутів), через те, що нова лінія починається у той момент, коли помилка перевищує поріг. Наведемо такий приклад: довга пряма лінія перетинає кут, а знехтувати кількістю (що залежать від порога) точок, які з’являються після перетину, можна раніше, ніж значення порогового рівня стане перевищеним.

Знайдену проблему можна усунути, якщо разом із методами об'єднання залучити методи розбиття.

Так, одним із методів розбиття сегментів межі є метод, у якому сегмент послідовно розподіляється на дві частини до того моменту, поки задовольняється заданий критерій.

1.8 Дескриптори областей зображень

1.8.1 Деякі прості дескриптори

Системи МЗ створені на значно простих дескрипторах області. Така база з обчислювальної точки зору надає їм більшої привабливості. Проте реалізація цих дескрипторів обмежена ситуаціями, де питомі об'єкти настільки відмінні, що їхня ідентифікація виконується за кількома основними дескрипторами.

Площу області можна обчислити числом пікселів у межах її кордонів. Такий дескриптор доцільно використовувати, щоб мати інформацію про взаємне розташування, форми об'єктів, від яких камеру обладнано на однаковій відстані. Так, за приклад візьмемо розпізнавання системою машинного зору об'єктів на рухомому конвеєрі.

Визначити орієнтацію об’єкта можна за допомогою великої та малої осі області. Ексцентриситет області, тобто відношення довжин цих осей, – це важливий дескриптор для опису форми області.

Довжина межі області називається периметром області. Проте периметр може виступати як дескриптор, наприклад, для визначення міри компактності області, отже, дорівнює квадрату периметра, поділеному на площу. Звернемо увагу, що компактність є безрозмірною величиною, відповідно, вона інваріантна до зміни масштабу, разом з тим є мінімальною для поверхні форми диска.

Область, в якій будь-яка пара точок може з'єднуватись кривою, повністю лежить в цій області називають зв’язковою. Дескриптором для

множини зв'язкових областей (деякі з них містять отвори) може бути число Ейлера. Мова йде про різницю між кількість зв'язкових областей та кількістю отворів. Так, числа Ейлера для букв *A* і *B* дорівнюють 0-1. Нижче наведемо приклади інших дескрипторів області.

1.8.2 Текстура

Ідентифікацію об'єктів або областей образу, здебільшого, можна виконувати дескрипторами текстури. Формально визначення текстури не існує, хоча інтуїтивно дескриптор такий можна трактувати як опис властивостей поверхні (однорідність, шорсткість, регулярність). До того ж, розрізняють два ключові підходи для опису текстури: статистичний і структурний. Перший метод (статистичний) відповідає за такі характеристики: зернистість, однорідність, шорсткість тощо. Другий метод (структурний) встановлює взаємне розташування елементарних частин образу. Наприклад, опис текстури, базується на регулярному розташуванні паралельних ліній [10].

1.8.3 Схема області

Для опису виду структури плоскої області використовують її представлення у вигляді графа. Для багатьох випадків необхідно розпочинати з визначення схеми (скелета) області. Ключовими на цьому етапі є проріджувальні (чи скорочувальні) алгоритми. Проріджувальні процедури є важливими для широкого діапазону завдань комп'ютерного зору: під час автоматичної перевірки друкованих плат або підрахунку азбестових волокон у повітряних фільтрах. За допомогою перетворення середніх осей (ПСО), запропонованого в роботі, можна визначити скелет області. ПСО окреслює досить задовільний скелет області, проте його пряме використання з обчислювальної точки зору є складним через необхідність обчислення відстані між межею та кожною точкою області. У зв’язку з цим запропоновано низку алгоритмів побудови середніх осей з більшою

обчислювальною ефективністю. Здебільшого, це алгоритми проріджування. Вони ітеративно усувають точки контуру області з опрацювання з метою виконання таких обмежень:

– не усувати крайні точи;

– не сприяти порушенню зв'язності;

– не викликати надмірне розмивання області.

1.9 Сегментація й опис тривимірних структур

Зір, фактично, – це тривимірна проблема. З огляду на це, основою розробки багатофункціональних систем МЗ, що можуть функціонувати в різних середовищах, покладено процес обробки інформації про тривимірні площини. Дослідження у цій галузі тривають вже понад 10 років. Однак чинники вартості швидкості разом зі складністю призупиняють упровадження обробки тривимірної зорової інформації до промислових застосунків.

Подавати інформацію про тривимірну площину можна трьома основними способами. За допомогою датчиків для вимірювання відстані отримуємо координати (*x, y, z*) точок поверхонь об'єктів [10]. Залучаючи пристрої, що відповідають за стереозображення, отримуємо тривимірні координати й інформацію про освітленість кожної точки. Отже, кожна точка представлена функцією *f (x, y,* z), де значення останньої в точці з координатами (*x, y, z*) переходять у значення інтенсивності в цій точці. До речі, позначати точки в тривимірному просторі та її інтенсивність можна терміном «воксель». Іншими словами, встановити тривимірні зв'язки можна, спираючись на один двовимірний образ площини. Це дозволяє виводити між об'єктами такі зв’язки, як «над», «за», «перед». З огляду на те, що на основі одного зображення, як правило, не може обчислюватись точне тривимірне розташування точок площини, зв'язки, отримані за результатами цього виду аналізу, іноді належать так званий 2,5-мірній інформації.

1.10 Опис тривимірної площини плоскими ділянками

До одного з елементарних підходів щодо сегментації та опису тривимірних структур за допомогою координат точок *(x,y,z)*, за потреби, можна додати розділення площини на невеликі плоскі «ділянки». У подальшому їх можна об'єднати на більші елементи поверхні за певними критеріями. Такий метод корисний для ідентифікації багатогранних об'єктів, у яких поверхні є гладкими щодо роздільної здатності.

Коли площину задано вокселем, її можна описати плоскими ділянками, послуговуючись тривимірним градієнтом. Таким чином, за результатами об’єднання цих плоских ділянок з’являються дескриптори поверхні. Вектор градієнта зазначає напрямок максимальної швидкості зміни функції, тоді як його величина відповідає величині цієї зміни. Такі риси характерні для тривимірного випадку, хоча можуть застосовуватись для розбиття на сегменти тривимірних структур за аналогією до двовимірних даних.

1.11 Техніка обробки візуальної інформації

Форма і розмір. Усі об’єкти, задіяні в системі автоматичного складання, вважатимемо точними проекціями відповідних реальних об'єктів. Вони можуть мати будь-яку форму та налічувати необмежену кількість. Будь-яке виявлення аналізується системою відхилення від розміру чи форми, тому може сприяти появі помилки під час збирання, служити підставою для відбраковування виробів (хоча перевірку виробів 100-відсотково проведено не буде). Отже, опис об’єкта, отриманий за результатами аналізу, повинен містити всю інформацію про розміри та форми цього об'єкта. Проте система розпізнавання символів у загальному випадку може послуговуватись значними відхиленнями (різне накреслення) в розпізнаваних символах [10]. Крім того, в найскладнішому випадку розпізнаються рукописні символи, хоча кількість типів розпізнаваних символів завжди обмежується. Найкраще

розпізнавання отримуємо під час застосування шаблону для аналізу друкованих шрифтів, які мають спеціальне призначення, зокрема, для оптичного зчитування, на кшталт, ОКР-А й ОКР-Б (OCR-A та OCR-B).

Параметри стану. Робот-складальник працює з деталями, що можуть розміщуватись у будь-якій точці в межах поля зору та довільною орієнтацію. У зв’язку з тим, що опис деталі виконується для подальшого її розпізнавання та за інваріантним до параметрів положенням, то виявлення значень вказаних параметрів теж вагома функція аналізатора візуальної інформації. Під час реалізації операцій використовують значення параметрів положення. Задачу щодо розпізнавання символів можна спростити. Однак рядки символів можуть бути нахиленими чи до поля зору можуть потрапити окремі символи. Здебільшого системи розпізнають у доволі жорстко фіксованому положенні, необхідності вимірювати параметри якого немає.

Кількість об'єктів. Вимагати від зорового аналізатора робота-складальника навичок працювати одночасно з декількома деталями, що знаходяться в полі зору та розташовані випадковим чином, – це закономірно. Розв'язання завдання, що потребує розподілу об'єктів під час розпізнавання символів, полегшується, якщо попередньо вже відомі розміри та курсив символу (в разі розпізнавання друкованих символів). Наявна інформація дозволяє використати прийоми поділу дотичних символів [10]. У сукупності їхній силует сприйматиметься як єдиний, найімовірніше, невідомий об'єкт.

У даній роботі завдання щодо розрізнення дотичних об'єктів не ставиться.

Відсутність обмеження щодо кількості типорозмірів об'єктів, із якими система розпізнавання працює безпосередньо, спричиняє необхідність у виконанні повного опису форми для кожного образу. Однак однозначно визначити контури будь-якої конфігурації можна за допомогою кількох ключових параметрів. Беручи їх за основу, отримуємо необхідну інформацію про тип і стан об'єкта.

Такими параметрами можуть бути: периметр, площа, мінімальний охоплюючий прямокутник (розміри), центр площі (центр ваги), радіус-вектор мінімальної та максимальної довжини (напрям та довжина), характеристики отворів (кількість, розмір, розташування).

Параметри площі та периметра – це найелементарніші характеристики інваріантні до орієнтації та розташування. Під час розв'язання задачі розпізнавання об'єкта застосовували безрозмірний «коефіцієнт форми» у значенні площа/периметр. Координати вершин, охоплюючого мінімального прямокутника, містять деяку інформацію про форми та розміри об'єкта, хоча їхнє значення залежить від його орієнтації. Точка, що легко визначається для будь-якого об'єкта незважаючи на місце його орієнтації, називається центром площі. Важливість центру площі виявляється під час виявлення та розпізнавання об'єкта. Крім того, від нього задається початок радіус-вектора, що з'єднує центр площі з точкою, котра розміщена на межі об'єкта. Здебільшого, радіус-вектори мінімальної та максимальної довжини можуть бути корисними під час розпізнавання та визначення параметрів положення об'єкта.

У промислових деталях, зазвичай, наявні отвори, кількість яких також характеризується як корисна особливість. Розпізнавати отвори можна як об'єкти певної форми та розміру. У свою чергу, вони орієнтовані на основний об'єкт, в якому були виявлені.

Найзручнішим способом для отримання докладного опису довільного геометричного об'єкта, а разом з тим і встановлення зв'язності образу вважається відстеження межі. Починається процес відстеження з вибору довільної точки, що розташована на чорно-білій межі проекції об'єкта. Далі реалізується алгоритм, який визначає суміжні точки, що знаходяться на межі, послідовно, поки не завершиться повний обхід контуру. Послідовно занотовуючи напрямки руху вздовж межі від кожної поточної точки до наступної, отримуємо одновимірний опис проекції об'єкта з інформацією про його форму. Ланцюжки напрямків вивчалися Фріменом всебічно [10]. За

результатами простеженої межі дуже просто визначити площу, периметр, центр площі та охоплюючий прямокутник. За описом межі також можна визначити радіус-векторів.

Відстеження межі виявляє зв'язність аналізованого об'єкта. Продовжуючи перегляди щодо пошуку наступних об'єктів, можна повторно знайти вже відстежену межу.

Щоб позбавитись такої дії, Розенфельд пропонує у всіх, переглянутих програмою відстеження межі, точках для першого оброблюваного об'єкта замінювати одиниці на двійки, для другого – на трійки, далі за такою ж послідовністю (одиниця співвідноситься з об'єктом або «чорним» елементом растра вихідного образу, нуль – фону або "білому" елементу). Зауважимо, що такий підхід розрахований для представлення кожної точки вихідного образу більше одного біта, відповідно, для систем із малим розміром пам'яті - непридатний.

Для розв’язання поставленого завдання доцільно залучати оператора поточкового диференціювання або виділення межі. Застосовувати слід його одразу після зчитування образу телекамерою. В образі оператор замінює всі одиниці на нулі, окрім точок, розташованих на межі чорного та білого полів. Контури квадрата та кола, що розташовані в полі зору камери, отже, отримані за допомогою ЕОМ, наведено на рисунку 1.2. Тут наведено результат застосування до цього зображення оператора виділення межі. Процедура відстеження межі EDGETRACE може працювати з проміжним зображенням за аналогією до вихідних об'єктів. Її побудовано за принципом: елементи растра, що вже переглянуті нею, «обнуляються». Таким чином, відстеження межі об'єкта видаляє його із зображення, при цьому повторно знайти його неможливо.

Згладжування. Спосіб обробки зображень об'єктів, який ґрунтується на відстеженні меж, може не спрацьовувати, зокрема, якщо на межі об'єкта спостерігатимуться різкі скачки контрастності. У системах розпізнавання символів відстеження меж зазвичай випереджається операцією згладжування

(локального усереднення), котра намагається знищити пробіли (білі крапки, що мають бути чорними), а разом з тим і помилкові чорні точки, до того ж, з'єднати невеликі розриви. В операції згладжування, за Дайніном, по черзі на кожен елемент растра накладається вікно розміром *nXn* елементів, а потім обчислюється кількість чорних елементів, які з’явились у вікні. Виникає нове растрове уявлення, де окремій позиції вікна відповідає кожен елемент. У такому растровому варіанті елемент буде чорним лише в тому разі, якщо кількість чорних елементів, які потрапили, відповідно, до вікна, перевищує деяку кількість, задану попередньо. Найменшим розміром вікна на практиці є 3×3 елемента. Свого часу Унгер запропонував процедуру згладжування, де замість усереднення вмісту вікна розміром 3×3 елемента до нього застосовують деякі логічні операції з метою виявлення значення елемента нового образу. Проте недолік розглянутих методів – виконання значного обсягу операцій із пам'яттю: в обох випадках усі обчислення значення елемента нового способу доводиться виконати, активізувавши до 100 машинних операцій.

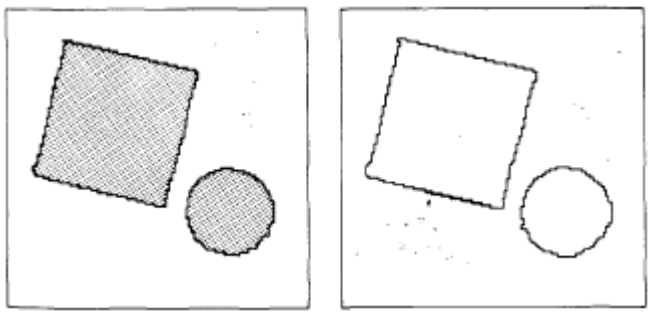

Рисунок 1.2 – Вміст растру після процедури INPUT-FRAME
і вміст растру після виділення меж

Об'єкти, з якими взаємодіє робот-складальник, на відміну від друкованих символів, якщо містять будь-які пробіли в відповідних їм образах, розглядаються як дефективні, а це є достатньою підставою для відбраковування такого об'єкта [10]. Подекуди виникають ізольовані шумові

точки. В цілому вони не впливають на роботу алгоритму відстеження межі. Відповідно, не використовуються спеціальні згладжувальні операції, але в процедурі відстеження заходи для зменшення впливу шумових точок можуть застосовуватись.

Виділення межі, іншими словами, процедура OUTLINEFRAME. Оператор виділення межі повинен забезпечувати безперервну послідовність граничних точок, реалізуючи можливість відстеження межі. Для мінімізації часу обчислення під час створення кожного елемента нового способу потрібно скористатися доступом до мінімально можливої кількості елементів вихідного образу.

Варіанти визначення контуру об'єкта. Розенфельд розмежовує поняття «край» і «межа» Край визначається як сукупність зовнішніх елементів растрового представлення об'єкта, межа, у даному разі, проходить посередині між горизонтально або вертикально суміжними точками, серед яких одна належить об'єкту, а інша – фону. У нашому випадку використаємо концепцію межі, бо вона точніше окреслює лінію, котра розподіляє чорні і білі області растра.

Зазначена вище концепція забезпечує однозначність рішення, за умови, що лінії товщиною в одну точку. Дотримання концепції краю призведе до результату, непридатного для подальшої роботи алгоритму відстеження, бо йдеться про збіг двох країв лінії.

Визначення межі за Розенфельдом має свій недолік: у підсумкового масиву точкова щільність вдвічі більша, ніж у вихідного за умови збереження взаємно однозначної відповідності двох масивів. Новий масив граничних точок створюється з аналогічною відстанню між точками, як і вихідний, однак точка зсувається за координатами x і y на половину цієї відстані. Кожна точка нового масиву знаходиться в центрі квадрата, в вершинах якого розташовані точки вихідного масиву, а його стан (чорне/біле, межа/не межа) окреслюється станом цих чотирьох точок.

Шість основних (із шістнадцяти можливих) поєднань станів чотирьох суміжних елементів вихідного образу. Крім того, наведені інтуїтивні певні значення відповідних елементів масиву з описом межі. Для конфігурації *d* вибираємо нульове значення, щоб у разі лінії з товщиною в одну точку, було забезпечено однозначність [10].

Відстеження межі, іншими словами, процедура EDGETRACE.

Параметри, що окреслюються процедурою EDGETRACE, представлено у таблиці 1.1.

Таблиця 1.1 – Параметри, визначувані процедурою EDGETRACE

Змінна	Значення
PERIMETER	Довжина відстежуваного контуру, позитивне ціле число
AREA	Площа, обмежена контуром, позитивне ціле число
XMAX XMIN YMAX YMIN	Максимальне і мінімальне значення координат точок, які належать контуру, негативні цілі числа
XCENTROID YCENTROID	Координати центру площі, обмеженою контуром, негативні цілі числа
CHA1NCOUNT	Кількість елементарних векторів, які утворюють контур, що збігається з кількістю переглянутих граничних точок, від'ємне ціле число
CHAIN	Упорядкований масив значень, задають напрямки елементарних векторів, які утворюють контур. Масив складається з елементів CHA1NCOUNT

EDGETRACE, іншими словами, процедура, що виявляє, чи належить межі об'єкта точка, виявлена в диференційованому способі. Координати початкової точки вводяться до глобальних змінних цілого типу STARTX, STARTY. Процедура EDGETRACE активізується як логічна процедура – функція, наприклад, в операторі

IF EDGETRACE THEN ... ELSE

Значення «істина» активується процедурою тоді, коли знайдено замкнуту межу, в якій розташовано більша кількість точок, ніж виявлено заздалегідь.

Процедура EDGETRACE окреслює формулювання сукупності параметрів, що належать до контуру. До того ж, створює опис самого контуру, подаючи його як список векторів, представлених глобальними змінними.

Визначення значень XM AX, XMIN, YMAX, YMIN уважається необов'язковим. Якщо у них виникає необхідність, доцільно встановити ознаку слова EDGEF-LAG (значення відповідного слова повинно стати ненульовим). За аналогією, використовуючи ознаку CHAINFLAG, можна активувати/деактивувати запис значень до лічильника CHAINCOUNT і масиву CHAIN.

Ланцюговий запис та основний алгоритм. У нашій роботі точка матриці, що співвідносна образу, вважається пов'язаною з іншою точкою, якщо остання розташована в одній з восьми позицій, суміжних з першою точкою безпосередньо.

З'єднання будь-якої пари пов'язаних точок можна подати одним із восьми елементарних векторів, напрямки яких мають значення від -8 до -1 (негативні значення потрібні для поєднання зі схемою індексації EOM PL-516).

Алгоритм відстеження межі «обнуляє» значення знайденої точки, що належить контуру, до переходу до наступної точки. Таким чином, з матриці FRAME видаляються відстежені окреслення. Базова точка пошуку переміщається контуром проти годинникової стрілки. Під час пошуку контурів за допомогою розгортки це забезпечує перше входження до контуру ззовні. Увійти зсередини до контуру можна лише тоді, коли він перетинає межу матриці. Разом з тим процедура EDGETRACE сформує повідомлення про помилку.

Необхідності перевіряти всі вісім сусідніх точок у кожній точці контуру об'єкта немає, тому що на попередньому кроці деякі з них уже були перевірені, бо спершу базовою точкою пошуку була попередня. Перед початком локального пошуку в залежності від напрямку попереднього вектора визначаються змінні SEARCHDIRN (для початку перевіряється напрямок) і ENDDIRN (останнє перевіряється напрямок).

На кожному кроці пошуку виникає до шести точок, проте, як правило, їхня кількість є меншою. На прямолінійних ділянках контуру до знаходження його наступної точки, здебільшого, достатньо переглянути одну-дві точки.

Повний опис окресленого контуру подається як список напрямків векторів, де кожна точка контуру з'єднується з наступною за тією послідовністю, як їх було знайдено (масив CHAIN). Результат реалізації процедури відстеження межі до простого контуру разом з отриманим «ланцюговим» описом продемонстровано на рисунку 1.3.

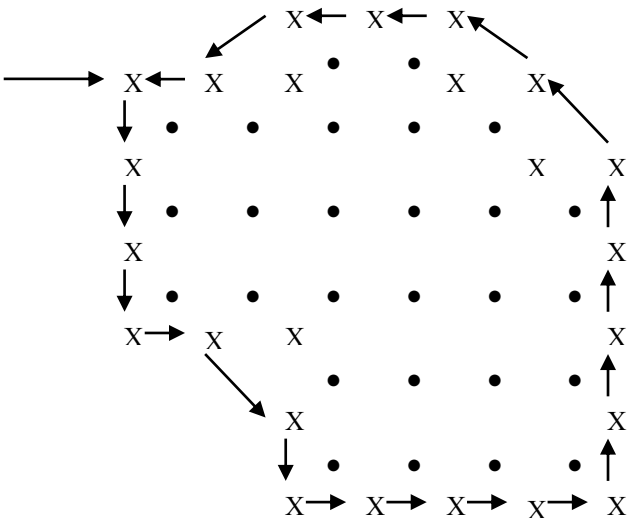

Рисунок 1.3 – Опис контуру простої форми

Водночас із пошуком точок контуру виконуються обчислення периметра, площі, а також моментів площі першого порядку [10].

Обчислення периметра. Суму довжин елементарних векторів, які складають контур, називають периметром. Такі вектори мають довжину 1

(напрямки парних номері) або одиниці растра (напрямки непарних номерів). Береться до відома, що відстань між точками в матриці повинна відповідати розміру елемента растра.

Із поступовим знаходженням нової точки контуру, з урахуванням напрямку останнього вектора, на одиницю збільшується значення однієї із змінних – EVENPERIM або ODDPERIM. Наприкінці процедури виконується обчислення периметра та його округлення до найближчого цілого значення:

PERIMETER = EVENPERIM + ODDPERIM

Обчислення площі. Обмежена контуром площа вираховується як сума площ, які обмежуються елементарним вектором, вертикалями, проведеними через його кінці, а також горизонтальною лінією (із значенням $Y = 0$). Якщо дотримуватись нумерації напрямків векторів, отримуємо, що елементарні вектори зі зменшувальною компонентою x (-5, -4, -3) вносять позитивні значення до загальної суми, а вектори зі зростаючою компонентою x (-7, -8, -1) – негативні (Y має негативне значення). За наведених умов обмежена контуром площа прослідковується проти годинникової стрілки з позитивним значенням.

Номери та значення ланцюжка продемонстровано у таблиці 1.2.

Таблиця 1.2 – Номери та значення ланцюжка

Номер елемента ланцюжка	Значення елемента ланцюжка
-1	-2
-2	-2
-3	-2
-4	-8
-5	-1
-6	-2
-7	-8
-8	-8

Продовження таблиці 1.2

Номер елемента ланцюжка	Значення елемента ланцюжка
-9	-8
-10	-8
-11	-6
-12	-6
-13	-6
-14	-6
-15	-5
-16	-5
-17	-4
-18	-4
-19	-3
-20	-3

З метою уникнення зайвих маніпуляцій з коефіцієнтом 1/2 процедура EDGETRACE накопичує подвоєне значення площі (AREA2), і ділить його навпіл наприкінці обчислень.

Формулювання перших моментів площі та центроїдів. Наведений вище метод характерний для обчислення моментів обмеженої площі першого порядку щодо осей *x* і *y*. Вісім простих векторів із першими моментами для площ, укладеними між віссю *x* (AW_X) і векторами. Сума всіх компонентів $ДЛ/_X$ для контуру сформує негативне ціле число, котре після поділу на значення обмеженої площі створить координату її центра по осі *y*.

Значення площ і прирощень подано у таблиці 1.3.

Таблиця 1.3 – Значення площ і прирощень

Напрямок вектора	Прирощення площі	Подвоєне прирощення
-8	Y	$2Y$
-7	Y-1/2	$2Y$-1
-6	0	0
-5	(Y-1/2)(-1)	-2Y+1
-4	Y(-1)	-2Y
-3	(Y+1/2)(-1)	-2Y-1
-2	0	0
-1	Y+1/2	2Y+1

Вираз для AM_X за наявності векторів із напрямками непарних номерів використовує постійний член 1/6 або -1/6. У замкнутому контурі, в якому присутня велика кількість простих векторів, кількість векторів усіх напрямків налічується приблизно однакова. В результаті чого отримуємо взаємне знищення постійних членів. До того ж, постійний член у порівнянні з Y^2 і з площею є доволі незначним (малим) особливо у тому разі, коли об'єкт, образ якого займає більшу частину поля зору, розглядається поряд [10]. Отже, якщо знехтувати постійним членом, то обчислення можна спростити (рисунок 1.4).

Операцій з коефіцієнтом 1/2 уникають за тим же принципом, що і під час обчислення площі, – виконується обчислення значення 2 $ДМ_X$. Для цього застосовують подвоєну довжину XMOMENT та процедуру SUMXMOMENT.

Під час завершального етапу процедури EDGETRACE XMOMENT ділиться на значення AREA2. За умови округлення результату до найближчого цілого отримуємо координату Y центру площі – YCENTROID.

Підсумовуючи також моменти площі щодо осі у набуваємо значення XCENTROID.

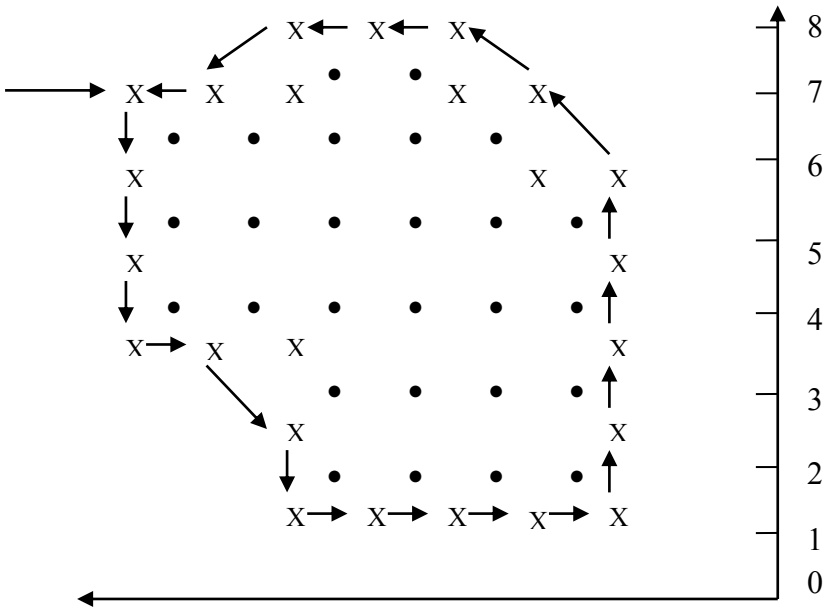

Рисунок 1.4 – Обчислення площі, обмеженою контуром

Одна чи незначна кількість суміжних "шумових" точок, наприклад, тих, які випадково потрапили до поля зору уламку стружки, утворюють контури, в яких є маленька кількість простих векторів. У такому разі значення для змінної MINCHAIN набуває -10. Іншими словами, будь-який замкнутий контур із довжиною не більше 10 векторів не розглядатиметься взагалі .

Обчислення сумарної площі подано у таблиці 1.4.

Таблиця 1.4 – Обчислення сумарної площі

Номер елемента ланцюжка	Значення елемента ланцюжка	Подвоєне прирощення	Подвоєна площа
-1	-2	0	0
-2	-2	0	0
-3	-2	0	0
-4	-8	-8	-8
-5	-1	-7	-15

Продовження таблиці 1.4

Номер елемента ланцюжка	Значення елемента ланцюжка	Подвоєне прирощення	Подвоєна площа
-6	-2	0	-15
-7	-8	-4	-19
-8	-8	-4	-23
-9	-8	-4	-31
-10	-8	0	-31
-11	-6	0	-31
-12	-6	0	-31
-13	-6	0	-31
-14	-6	+13	-31
-15	-5	+15	-18
-16	-5	+15	-3
-17	-4	+16	+13
-18	-4	+16	+29
-19	-3	+15	+44
-20	-4	+14	+58
Сумарна площа = 29			

Значення моментів відносно осі X і Y представлено у таблицях 1.5-1.6.

Таблиця 1.5 – Значення моментів відносно осі X

Напрям вектора	Прирощення площі	Подвоєне прирощення
-8	$Y^2/2$	Y^2
-7	$Y^2/2- Y/2+1/6$	$Y(Y-1)$
-6	0	0
-5	$- Y^2/2+ Y/2-1/6$	$Y(-Y+1)$

Продовження таблиці 1.6

Напрям вектора	Прирощення площі	Подвоєне прирощення
-4	$-Y^2/2$	Y^2
-3	$-Y^2/2+Y/2-1/6$	$Y(-Y+1)$
-2	0	0
-1	$Y^2/2-Y/2+1/6$	$Y(Y-1)$

Таблиця 1.6 – Значення моментів відносно осі Y

Напрям вектора	Прирощення площі	Подвоєне прирощення
-8	0	0
-7	$X^2/2+X/2+1/6$	$X(X+1)$
-6	$X^2/2$	X^2
-5	$X^2/2-X/2+1/6$	$X(X-1)$
-4	0	0
-3	$-X^2/2+X/2-1/6$	$X(-X+1)$
-2	$-X^2/2$	$-X^2$
-1	$-X^2/2+X/2-1/6$	$X(-X-1)$

Матриця образу після процедури EDGETRACE. Описаний алгоритм відстеження меж за умови оброблення диференційованої матриці обнуляє всі переглянуті точки, однак про те, що будуть переглянуті і обнулені всі точки контуру, мова не йде. Одиничні точки, що так і перебувають у матриці, як правило, повного контуру не становлять, отже, сприймаються як "шумові" під час подальших операцій. Для мінімізації їхнього впливу доцільно вжити особливі заходи. На рисунку 1.5 продемонстровано вид матриці образу FRAME після опрацювання процедурою EDGETRACE обидвох контурів. Бачимо, що шумові точки залишились.

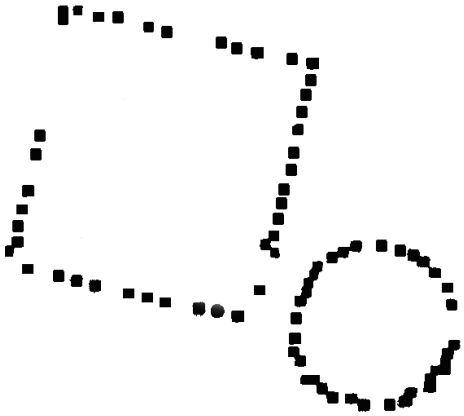

Рисунок 1.5 – Матриця зображення після відстеження обох меж

Ідентифікація отворів іншими словами процедура INSIDE. Процедура INSIDE визначає розташування деякої точки всередині або зовні певного контуру, що належить матриці образу FRAME [10].

Мається на увазі, що контур, виділений процедурою EDGETRACE, зберігається як масив CHAIN, а визначає його довжину змінна CHAINCOUNT. Початкову точку ланцюжка можна задати цілими змінними PARTXSTART та PARTYSTART.

Координати точки вводяться до змінних STARTX і STARTY і під час перевірки активується логічна процедура-функція INSIDE. Вона набуває значення «істина», якщо перевірна точка розташована всередині контуру. Зауважимо, що в пам’яті жодні значення не змінюються.

Розглянутий алгоритм базується на такій, справедливій для площині, теоремі: якщо точка *P,* що розміщена поза замкнутої кривої *S,* з'єднується відрізком прямої з точкою *Q,* то точка *Q* теж знаходиться поза кривої *S* лише у тому разі, коли відрізок *PQ* перетинає криву *S* парну кількість раз. А за умови, що *PQ* перетинає криву *S* непарну кількість раз, відповідно, точка *Q* знаходиться всередині замкнутої кривої *S* (рисунок 1.6). Такий доказ наводять Курант з Роббінсом.

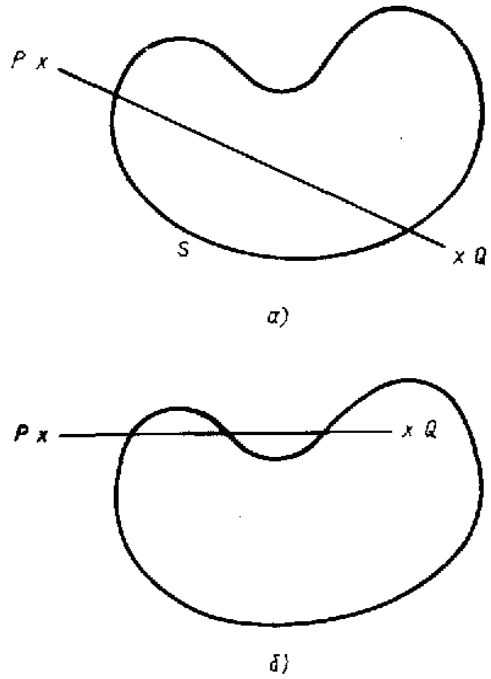

Рисунок 1.6 – До теореми про перетин прямої і замкнутої лінії

Пара локальних змінних (XCOORD та YCOORD) спершу визначаються як рівні координати початку зберігання контуру. Послідовно з масиву CHAIN зчитуються прості вектори. Модифікуються XCOORD і YCOORD в залежності від значення (процедура UNCHAIN). У зв’язку з цим XCOORD та YCOORD послідовно приймають значення координат усіх точок, які було прослідковано у вихідному контурі [10].

На кожному кроці обов’язково звіряється, чи не розміщено точку контуру на промені У = STARTY, x > STARTX. Кількість перетинів контуру з цим променем визначається в цілій змінній CROSSCOUNT. Не враховуються множинні перетину, якщо в кількох суміжних точках лінія контуру збігається з тестовим променем. У момент входження лінії контуру тестовий промінь у змінній STRIKEDIRN надається індекс напрямку вектора, котрий входить в промінь. Для решти точок, які розміщені на промені, не виконується жодних дій. У разі виходу наступного вектора контуру з тестового променя, виконується порівняння поточного напрямку зі значенням, яке записано у STRIKEDIRN для перевірки: чи перетинав

тестовий промінь контур (CROSSCOUNT збільшується на одиницю) чи контур рухався відносно променя (CROSSCOUNT не змінюється).

У разі збереження всіх точок контуру, реалізується процес відновлення та перевірки, зокрема, значення CROSSCOUNT. Завершується процедура з відповідним результатом. Неоднозначні граничні випадки, за умови, що перевірені точки розташовані на самому контурі, усуваються, бо в процесі відстеження межі точки зазнають обнулення.

Як контури об'єкта вже виявлено, переходимо до визначення його орієнтації разом з положенням. Отримання такої інформації сприяє правильній роботі з деталлю в процесі складання.

Завдання визначення орієнтації. Основні процедури щодо обробки візуального образу виконуються з контурами (об'єкта й отворів у ньому), параметрами площі, периметра, положення центру площі, що охоплюють прямокутник, описом контуру ланцюжком векторів. Проте вони актуальні лише для початкової обробки. Для процесу розпізнавання образу, а також визначення його орієнтації доцільно обробляти вже інформацію про контури й отвори.

Модель об'єкта з отворами. Модель, утворену із отворів, наявних в об'єкті, можна описувати, ґрунтуючись на відстані центрів площ отворів від центру площі об'єкта та їхніх відносних кутових координат. Водночас послуговуватись інформацією про вже виміряні площі та периметри отворів.

Модель, яка стає інваріантною до орієнтації основного об'єкта, безпосередньо може порівнюватися з моделлю-зразком. Значення абсолютних кутових координат моделі можна застосовувати для визначення орієнтації об'єкта після того, як попередньо вже встановлено відповідність між окремими отворами розглянутого об'єкта та моделлю.

Виділення особливостей контуру. Модель контуру може бути представлена як список–ланцюжок векторів. Проводити порівняння вхідної інформації, представленої такою моделлю, зі зразком, як Фрімен це робить, насправді, довго і складно. Складність, зокрема, ґрунтується на тому, що

початкова точка ланцюжка, що відповідає контур об'єкта, на ньому розташована випадковим чином. Її конкретне положення підпорядковується орієнтації об'єкта та напрямкам сканування під час перегляду. Ланцюжки, що співставляються, доводиться циклічно зміщувати відносно один одного до збігу. Наголосимо, що опис контуру ланцюжком векторів не зовсім вигідний для проведення операції повороту на кут, не кратний 45º. Встановлення відповідності значно ускладнюється у зв’язку з тим, що зразок та вхідний ланцюжок у загальному випадку складаються елементарних векторів різної кількості.

Опрацьовано низку методів згладжування ланцюгових списків і нормалізації їхніх довжин, які сприяють порівнянню. Зах і Монтанарі описали методи отримання апроксимацій, застосовуючи прямі з мінімальною довжиною. В результаті обсяг даних, які підлягають обробці, знижується. Зауважимо, що в разі застосування в системі автоматичного складання згадані методи виявляться занадто складними. Ключова вимога системи – це не забезпечення 100 % збігу з еталонами у ході порівняння моделі контурів, виокремлених із вхідного зображення, а підтвердження результатів пробних перевірок та окреслення орієнтації об'єкта.

Потрібно обрати модель, яка дозволить якомога легко та швидко окреслити орієнтацію об'єкта. Більш за всіх для такої моделі можна застосувати підхід, який базується на формулюванні особливостей описаного контуру. Декілька методів розпізнавання символів, на кшталт Хоскінга, застосовують повторювані характеристики структури штрихів, створюючи символ. До згаданих особливостей належать: кінці штрихів, петлі, точки їхнього з'єднання чи зламу. Подібні особливості, наявні у матеріальних об'єктах, можуть виражатися різкими змінами напрямку контуру (кути) чи точки, що віддалені від центру площі фігури як максимально, так і мінімально. Відсутність обмежень щодо форми виробів, з якими взаємодіє робот-складальник, применшує цінність методів, які раніше використовував складений набір стандартних особливостей.

Модель із використанням кіл. Кутові координати радіус-векторів мінімальної та максимальної довжини можуть бути зручним критерієм як для розпізнавання об'єкта, так і для визначення його орієнтації. Значення їхнє обчислюється з інформації, що генерується процедурою EDGETRACE безпосередньо. Проте якщо в околі екстремально віддаленої від центру площі точки мало змінюється довжина радіус-вектора, координати R_{MAX} *та* R_{MIN} стають ненадійними. У такому разі під час налаштування системи більш доцільно застосовувати варіант, в якому задаватимуться такі значення довжини радіус-вектора R, де може бути чітко визначеною відповідна позиція кінцевої точки. Іншими словами, це призводить до накладання заданого радіуса на образ кола. При цьому центр його співпадає з центром площі об'єкта. Точки перетину цієї окружності та контуру об'єкта подаються як його особливості. Параметрами R, 0, $в_2$, $в_3$, $в_4$ визначаються особливі точки. Різниці кутів – це інваріантний до орієнтації набір параметрів для розпізнавання, чиї абсолютні значення можуть застосовуватися для вираховування орієнтації об'єкта.

Точки перетину визначаються на базі інформації, наданою процедурою EDGETRACE: для кожної точки контуру, представленого ланцюговим списком векторів, її відстань обчислюється від центра площі. Окреслюються точки, в котрих ця відстань дорівнює заданому радіусу, та обчислюються їхні кутові координати відносно центру площі. В результаті точки перетину опрацьовуються у вигляді впорядкованого списку. З якої саме точки розпочато зазначений список – залежить від того, яке місця контуру було стартовим для процедури EDGETRACE.

За критерії щодо вибору значення радіуса кола візьмемо такі положення:

− точки перетину повинні однозначно визначати орієнтацію об'єкта;

− кількість точок перетину не повинна бути дуже великою (наприклад, не більше восьми);

− для підвищення точності вимірювань радіус, за можливості, повинен бути великим;

− незначні зміни довжини радіуса не створюють суттєвого впливу на кутові координати точок перетину;

− незначні зміни довжини радіуса не повинні спричиняти зникнення чи появу нових точок перетину.

Прикінцеві два пункти необхідні для зниження впливу незначних спотворень у вхідному поданні образу.

Щоб визначити орієнтацію деталі точно або відрізнити її від іншої схожої, може знадобитися більше однієї окружності. Тоді точки перетину, знайдені шляхом обходу контуру, подаватимуться як впорядкований список зі значеннями пар радіус-кут.

Порівняння значень виміряних кутів стає необхідним, якщо виникає повний збіг списків позначень радіусів. Процедура порівняння спрощується. Встановивши відповідність обох списків, можна вираховувати різниці абсолютних кутових координат для кожної пари суміжних точок перетину. Шість значень задіяно для визначення орієнтації одного об'єкта щодо іншого. Наступною властивістю моделі з використанням кіл вважається можливість фіксувати випадок, коли деталь лежить зворотною стороною до гори. Таким чином, розглянута модель – потужний засіб визначення особливих точок на описаному контурі об'єкта довільної форми. За допомогою таких точок реалізується процес розпізнавання та визначення орієнтації. Обсяг пам'яті, необхідний для зберігання моделі-еталона, на кожну точку перетину містить всього два слова, а дані про скановані деталі можуть надходити з інформації, що генерується процедурою EDGETRACE. Разом із моделлю, в якій є отвори, вона реалізується на нижчому рівні розпізнавання та визначення орієнтації виробів у системі керування робота-складальника.

2 МЕТОДИ ІДЕНТИФІКАЦІЇ ВИРОБІВ У РОБОТИЗОВАНИХ СИСТЕМАХ

2.1 Метод порівняння з еталоном

Ідентифікація об'єктів у машинному зорі найчастіше виконується методами порівняння з еталоном. При цьому машинний зір розв’язує одне з двох завдань. Перше полягає в отриманні зображення одного об'єкта, котре слід порівняти з усіма зразками заданого класу. Найкращий еталон обирається за збігом, потім проводиться ідентифікація об'єкта, після чого, за необхідності, обчислюються параметри його положення й орієнтації [11].

Друге завдання ґрунтується на отриманні зображення декількох об'єктів та почерговому їхньому порівнянні з еталоном об'єкта, котрий необхідно виділити. Потім проводиться обчислення його положення й орієнтації в робочій зоні робота.

Метод базується на встановленні збігу двох точкових зображень. Якщо збіг виявлено, то предмет, відповідно, розпізнано. Метод накладення еталона переважає над іншими, маючи незалежність від конкретного виду та складності об'єкта. За наявності двовимірних систем легко розпізнати ключ, як і коло чи будь-яку іншу фігуру. Хоча на згаданий метод розпізнавання можна витратити з багато часу, якщо застосовувати його до обчислювальної машини. На прикладі продемонструємо ймовірні труднощі, зокрема, контури, відповідні кубу і піраміді, зображено на рисунку. Як приклад, ставиться завдання визначити, чи містить де-небудь ця фігура квадрат, застосовуючи метод збігу зображень. Спроєктуємо еталон з чорних клітин, які сформують квадрат, оточений зовні та всередині білими клітинами. Накладемо його на фігуру та рухатимемо доти, поки збіг не встановимо. Однак можуть існувати й труднощі:

− першопочатково еталон проєктується правильного розміру, бо в іншому випадку необхідно створити безліч еталонів, які відрізнятимуться лише масштабним збільшенням;

– передбачуваний квадрат може знаходитись в будь-якій, невідомій нам, частині фігури. У зв’язку з цим пошук необхідно проводити як за зрушенням, так і за обертанням;

– позначається просторове квантування. Навіть за умови правильного накладення еталона збіг не буде ідеальним, а найкращим із можливих. Метод порівняння з еталоном для роботів доцільний лише з точки зору наявності інформації, що дозволяє заздалегідь обчислити розмір еталона, до того ж, привести кількість порівнянь до реально можливого виконання.

Метод порівняння з еталоном корисний в окремих випадках. З огляду на те, що способи виготовлення оптичних кореляторів (і навіть електронних кореляторів як дисектор зображень) є актуальними, попередньо метод застосовувався в оптичних пристроях, якими роботи не були обладнані. Проте розміри використовуваних еталонів завжди відомі наперед, а предмети розставлені й орієнтовані належним чином. Залишається розв’язати завдання щодо розпізнавання об'єкта із заданої множини об'єктів. Робот повинен розв’язувати більш складні завдання.

Здебільшого не досягається повного збігу об'єкта з еталоном у просторі вибраних ознак. Отже, вводиться допустиме розходження між еталоном і зображенням, відповідно перевіряється їхній збіг у межах зазначеної відмінності.

Брагін і Войлов [5] висувають таке трактування даного методу: якщо початкове зображення об'єкта позначити як $F(i, j)$, еталон $E(j, i)$, а обчислену різницю T, то процедура порівняння набуває вигляду

$$T = \sum_i \sum_j | F(i, j) - E(i, j) |^2 , \qquad (2.1)$$

$$T = \sum_i \sum_j | F(i, j) - E(i, j) | \quad . \qquad (2.2)$$

Індекси *i* та *j* характеризують положення елементів у дискретному цифровому зображенні.

Збіг еталона з об'єктом перевіряється правилом:

$$D \geq T, \qquad (2.3)$$

де *D* – задане граничне розходження.

У разі не виконання умови слід еталон замінити чи перейти до іншого зображення.

Детальніше розглянемо особливості методу порівняння з еталонами при використанні деяких систем ознак. Якщо за показники обрані площа і периметр зображення, розміри вписаних та описаних фігур, момент інерції і подібні геометричні властивості, то необхідно звертати увагу на масштабування, а на етапі попередньої обробки за якимось параметром нормувати зображення. Наведемо приклад, де площа описаного прямокутника або кола нормується квадратом периметра зображення, а периметр – значенням кореня квадратного з площі зображення [11].

Зображення акумулятивної різниці формується в результаті порівняння еталонного образу з кожним образом даної послідовності. Під час побудови зображення акумулятивної різниці використовують лічильник, призначений для обліку розташування пікселів. Його значення кожного разу збільшується, якщо виникає відмінність у розміщенні відповідних пікселів еталонного образу й образу з розглянутої послідовності. Отже, коли реалізується порівняння *k-го* кадру з еталонним, запис у цьому пікселі акумулятивної різниці показує, у скільки разів інтенсивність пікселя *k-го* кадру відрізняється

від інтенсивності пікселя еталонного образу. Відмінності можна встановити, розв’язавши рівняння.

Виникають ситуації, коли корисно опрацювати три типи зображень акумулятивної різниці: абсолютне, позитивне і негативне.

Позитивний результат застосування методів залежить від еталонного образу, з яким реалізуються подальші порівняння. Повторимо, що відмінність між двома образами в завданні щодо розпізнавання рухомих виробів виявляється шляхом виключення стаціонарних компонент за умови збереження елементів, які співвідносяться з шумом та рухомими виробами. Проблема виокремлення образу з шуму розв’язується методом фільтрації чи за допомогою формування зображення акумулятивної різниці.

На практиці отримати еталонний образ, щоб він мав тільки стаціонарні елементи, не завжди виходить, а це спричиняє потребу у побудові еталона з набору образів, що налічують один або більше рухомих виробів. Це, здебільшого, характерно для ситуацій, що описують площини з багатьма швидкоплинними об'єктами або у разі частих змін площин. Звернемо увагу на процедуру генерації еталонного образу. Розглянемо перший образ послідовності як еталонний. Наприклад, нестаціонарна компонента повністю вийшла зі свого становища в еталонному кадрі, тоді відповідний фон даного кадру можна перенести в положення, займане об'єктом в еталонному кадрі на початку. Коли всі рухомі об'єкти повністю покинуть свої першопочаткові положення, за результатами цієї операції відтворюється еталонний образ з наявними лише стаціонарними компонентами. Переміщення об'єкта реалізується операцією розширення позитивного зображення акумулятивної різниці.

2.2 Метод теорії графів

Під час визначення країв і контурів зображень застосовують методи графів. Дослідимо глобальний підхід, в якому у вигляді графа подаються

сегменти контуру і пошуку на графі шляху найменшої вартості, який відповідає значущим контурам, поданий у книзі Брагіна і Войлова [5]. Власне підхід є наближеним методом, ефективність якого діє при наявності шуму. На практиці ця процедура значно складніша, тому вимагає більше часу обробки, ніж методи, опрацьовані вище.

Для початку розглянемо кілька простих дефініцій. Граф $G = (N, A)$ – це кінцева, непорожня множина вершин N разом з множиною A невпорядкованих пар різних елементів з N. Кожна пара з A створює дугу.

Граф зі спрямованими дугами називається спрямованим. Якщо дуга виходить з вершини n_i, до вершини n_j, тоді n_j іменується наступником вершини n_i. В цьому випадку вершина n_i визначається як попередник вершини n_j. Процес ідентифікації наступників кожної вершини йменується розширенням цієї вершини. У кожному графі рівні визначаються за принципом, щоб нульовий рівень мав тільки одну вершину – початкову, а останній рівень містив вершини цільові. Кожній дузі (n_i n_j) приписується значення с (n_i n_j). Послідовність вершин n_1, n_2, ..., n_k, в якій кожна вершина n_i – наступникк вершини r_{i-1}, розуміється як шлях від n_i до n_k, а його показник обчислюється за формулою

$$c = \sum_{i=2}^{k} c(ni-1, nj)\,. \qquad (2.4)$$

Елемент контуру доцільно визначити як межу між двома пікселями p та q. У нашому випадку під контуром мається на увазі послідовність елементів контуру.

Важливим методом ідентифікації зображень з геометричними або іншими показниками вважається метод побудови графів рішень. Його успішно залучають у тих випадках, коли в заданому класі зображень є об'єкти, котрі розрізнити за однією ознакою зображення неможливо, тому для правильного розпізнавання потрібно використати кілька ознак. Від методу порівняння зображення й еталона за вектором ознак метод графів

відмінний тим, що на кожному його етапі порівняння відбувається відбір можливих рішень. Отже, кількість можливих варіантів розв'язання задачі розпізнавання на кожному етапі порівняння зменшується.

Граф (або дерево) розпізнавання за геометричними показниками зображено на рисунку 2.1.

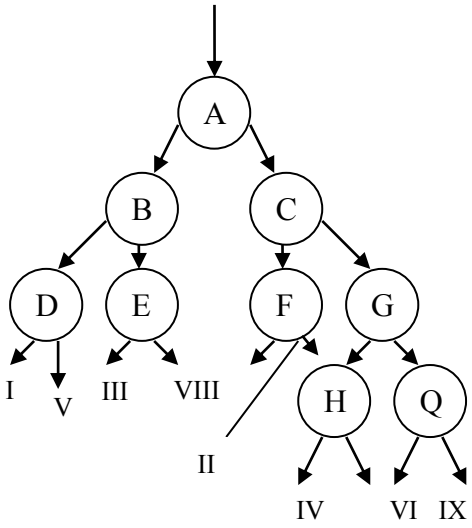

Рисунок 2.1 – Дерево розпізнавання

Цифрами I, II, …, *X* позначено можливі рішення – номера розпізнаваних виробів. Літерами *A*, *B*, …, *Q* оператори позначають вершини графа, що виділяють певні ознаки зображення. Наприклад, оператор *A* класифікує зображення за довжиною та висотою описаного прямокутника, оператори *B* і *C* – за площею. Оператори *DEFG* можуть проводити класифікацію за кількістю кутів, *H* і *Q* – за розташування кутів один від одного. Граф може мати як більше, так і менше рівнів, зміст операторів теж може бути різним.

2.3 Метод кореляції

Модифікація методу порівняння з еталоном реалізується через кореляційний метод, що ґрунтується на обчисленні взаємокореляційною функцією між еталоном і зображенням.

Кореляцію визначають як статистичний взаємозв'язок двох або декількох випадкових величин (або величин, які можна вважати такими, бо наділені деяким допустимим ступенем точності). Зауважимо, що зміни однієї чи декількох із цих величин призводять до систематичної зміни іншої/інших величин. Математичної мірою кореляції двох випадкових величин стає коефіцієнт кореляції.

Метод обробки статистичних даних, який ґрунтується на вивченні коефіцієнтів кореляції між змінними, називається кореляційним аналізом. У ньому порівнюються коефіцієнти кореляції між однією парою або множиною пар ознак з метою встановлення статистичних взаємозв'язків між ними.

Завдання кореляційного аналізу – забезпечення отримання деякої інформації про одну змінну за допомогою іншої. У разі, коли досягнення мети є можливим, кажуть, що змінні корелюють. У загальному вигляді прийняття гіпотези щодо наявності кореляції означає, що зміна значення змінної *А*, проводиться одночасно з пропорційною зміною значення *Б*.

Класифікація зображень здійснюється за результатом: чим більшим буде значення функції взаємної кореляції, тим з більшою ймовірністю еталон збігатиметься із зображенням. Застосовуючи позначення, прийняті у виразі, представимо формулу для обчислення взаємокореляційної функції *К* як:

$$K = \sum_i \sum_j F(i, j)\, E(i, j)^2 \,. \qquad (2.5)$$

Максимальне значення взаємокореляційної функції дорівнюватиме,

$$\sum_i \sum_j | E(i, j) |^2, \tag{2.6}$$

і досягатиметься за умови повного збігу зображення з еталоном. Нормована взаємокореляційна функція

$$R = \frac{\sum_i \sum_j F(i, j) E(i, j)}{\sum_i \sum_j | F(i, j) |^2}, \tag{2.7}$$

за умови збігу еталона із зображенням досягатиме максимального значення, що дорівнюватиме одиниці.

Застосування кореляційного методу та методу прямого порівняння з еталоном висуває загальні вимоги до процесу попередньої обробки зображень. Зокрема, зображення та еталон повинні бути однаково орієнтовані, мати рівний масштаб і зсуватися один до одного в полі зображення. Враховуємо й іншу властивість цих методів – необхідність застосування великої кількості еталонів.

Вони стають актуальними, коли завдання щодо розпізнавання виробів розв’язуються шляхом зміни їхньої проекції.

2.4 Розпізнавання через зв'язок шаблонів

2.4.1 Пошук виробів зазначенням зв'язків між шаблонами

Як правило, відстежуваний об'єкт наділений внутрішніми ступенями волі, отже, його зовнішній вигляд може сильно варіюватися (наприклад, люди рухають руками і ногами, риби деформуються під час плавання, змії звиваються тощо). Таке явище може надзвичайно ускладнювати процес

порівняння з шаблоном, оскільки виникає потреба або в класифікаторі з гнучкими межами (і безліч зразків), або багато різних шаблонів.

Багато виробів зазначеного типу складаються з невеликої кількості строго впорядкованих компонентів. Можна спробувати узгодити дані компоненти як шаблони, після чого виявити, які об'єкти присутні, проаналізувавши запропоновані зв'язки між знайденими шаблонами. Як приклад: пошук особи можна виконувати не за одним повним шаблоном, шукати окремо очі, ніс і рот з прийнятним взаємним розташуванням.

У даному підході можемо назвати кілька потенційних переваг. Перша: дізнатися шаблон «очі» можна легше, ніж шаблон «особи», тому що перша структура, вочевидь, є простішою. Друга: можна отримувати та застосовувати відносно прості ймовірнісні моделі, оскільки можуть існувати деякі властивості незалежності, на які можна спиратися. Третє: ймовірно, вдасться узгодити велику кількість виробів з відносно невеликою кількість шаблонів. Влучним прикладом цього явища може бути обличчя тварин. Переважна більшість тварин із характерними обличчями мають очі, ніс та рот, відмінність полягає лише в просторовому розташування цих елементів. Таким чином, будувати складні вироби можна на простих окремих шаблонах. Іншими словами, люди можуть рухати руками і ногами, а навчити цілісний явний шаблон виявляти людей вцілому, схоже, значно складніше, ніж отримати окремі шаблони для частин тіла, а також вірогідну модель, що описує їхні ступені свободи.

Розглянута тема наразі не настільки добре вивчена, щоб її стосувався якийсь стандартний підхід. Водночас досить очевидним є головне питання: як закодувати набір зв'язків між шаблонами в форму, з якою елементарно працювати. В даному розділі проаналізуємо низку різних підходів до поставленої задачі. По-перше, кожен шаблон може вказувати на об'єкти, котрі він може представляти, а потім певним чином обчислюється число покажчиків. Якщо створити деяку явну вірогідну модель, то для опису виробів просторових відношень можна застосувати більше вагових

коефіцієнтів. Таку модель можна побудувати з функцій правдоподібності, зокрема, потрібна функція розподілу ймовірностей, яка відіграє велике значення, коли конфігурація компонентів подібна об'єкту, а в іншому випадку – незначне. У такому разі пошук виробів перетворюється на пошук шаблонів, які за умови підстановки до вірогідної моделі набувають великих значень. Зауважимо, що слід ретельно ставитися до скорочення пошуку. Складність цього підходу полягає в тому, що навіть за умови скорочення пошук може бути вартісним. За твердженнями Форсайта і Понса, водночас при певному класі вірогідних моделей можна виконати ефективний пошук.

Прості моделі виробів спроможні забезпечити достатньо ефективне розпізнавання. Найпростіша модель – це розгляд об'єкта як набору фрагментів зображення (невеликих країв елементів характерного опису) кількох різних типів, які формують образ (pattern). Щоб визначити, який образ власне відстежується, знаходяться всі фрагменти, кожен з яких вказує на всі образи, до котрих він належить. Те зображення, на якому знайдено найбільшу кількість, і вважається затребуваним. Незважаючи на простоту дана стратегія є доволі ефективною. Нижче розглянемо методи пошуку фрагментів, а потім визначимо низку реалізацій із поступовим ускладненням даної стратегії.

2.4.2 Опис фрагментів зображення

Невеликі фрагменти зображення можуть мати доволі характерний вигляд, якщо налічують багато ненульових похідних (наприклад, в кутах). На зображенні автори [6] шукали кути, ще їх називають точками інтересу. Потім оцінювали набір похідних функції яскравості в зазначених кутах, а також набір похідних, які є інваріантними щодо обертання, трансляції, змін визначеного масштабу разом із освітленням. Такі ознаки йменуються інваріантними локальними особливостями (invariant local jets).

Припустимо, що фрагменти зображення можна розподілити на кілька класів. Представники кожного класу можуть бути сформрвані шляхом

використання декількох зображень кожного об'єкта, зазвичай, відповідні фрагменти заноситимуть до одного класу, проте внаслідок шуму можна отримати дещо відмінні інваріантні локальні потоки. Відповідний набір класів можна виокремити чи за допомогою ручної класифікації фрагментів, чи шляхом кластеризації фрагментів-еталонів (ліпший метод). Далі слід сформулювати умови, за яких два набори інваріантних локальних потоків відображають один клас фрагментів зображення. Для перевірки науковці Шмід (Schmid) і Мор (Mohr) застосовували відстань Махаланобіса (Mahalanobis distance) між векторами ознак протестованого фрагмента та фрагмента-еталона (якщо замір менший деякого порога, то протестований фрагмент уважається ідентичним до еталона.

Зауважимо, що по суті описаний класифікатор розподіляє фрагменти між класами, представленими еталонами чи відмовляється від класифікації, а фрагменти стають шаблонами. Програму щодо узгодження шаблонів можна побудувати за допомогою будь-якої техніки, без залучення ознак, які є інваріантними до обертання, як підкреслювали Форсайт і Понс [6]. Для цього можна використати набір налаштувань, в якому наявні версії кожного фрагмента-еталона, що відрізняються обертанням і зміною масштабу, до того ж, варіюються за умов освітлення так, щоб класифікатор міг виявити, що обертання, зміна масштабу і освітлення не чинить вплив на ідентифікацію фрагмента. Перевагою щодо використання інваріантних ознак назвемо таку, що класифікаторам не потрібно дізнаватися про інваріантності з набору налаштувань.

2.4.3 Вказівка показників і проста породжувальна модель

Припустимо, існує зображення, в якому розташовані точки, цікаві для нас, і в кожній точці зображення класифікується його фрагмент. Виникає питання: які ж образи розміщено в зображенні? Для відповіді на це питання можна побудувати відповідність між образами та фрагментами зображення. Нехай все на зображенні – це N_i фрагментів. Крім того, припустимо, що на

зображенні присутній або один образ з даної колекції, або жоден. Окремий фрагмент може породжуватися або єдиним наявним образом або шумом. Проте, образи, здебільшого, містять фрагменти не всіх класів. Таким чином, твердження щодо присутності певного способу рівнозначно твердженню про те, що деякі фрагменти зображення породжені шумом (у зв'язку з тим, що присутнім може бути тільки один образ, однак існують фрагменти зображення, що належать класам, які відсутні в поточному образі).

У результаті отримуємо просту породжувальну модель зображення. Коли наявний образ, можуть виникнути тільки фрагменти деяких певних класів. Створюючи таку модель, отримуємо низку алгоритмів зіставлення образів із відстежуваними зображеннями.

Найпростіша версія даної моделі ґрунтується на припущенні, що образ породжує всі фрагменти класів, які він може породити, тобто шумом має породжуватися мінімальна кількість фрагментів. Таке припущення зводиться до простої схеми вказівки. На кожному фрагменті зображення вказується кожен образ, який можуть породити фрагменти цього класу. Переважає спосіб, який отримав найбільшу кількість вказівок. Іншими словами, він вважається присутнім на зображенні. Наведена стратегія може бути ефективною, втім її застосування призводить до деяких проблем.

2.4.4 Імовірнісні моделі для вказівки

Простий процес рішення можна інтерпретувати як імовірнісну модель, оскільки при цьому окреслюються сильні та слабкі сторони описаного підходу. Фосайт і Понс [6] наголошують, що породжену модель можна перетворити на ймовірнісну, припустивши, що генеруються фрагменти незалежно і випадковим чином, до того ж передбачено присутність об'єкта. Запишемо

$$P\{\text{фрагмент } i\text{-го типу є на зображенні} \mid \text{присутній } j\text{-й образ}\} = p_{ij}$$
$$P\{\text{фрагмент } i\text{-го типу} \mid \text{немає образу}\} = p_{ix}.$$

У простій моделі передбачається, що для кожного образу j, $p_{ij} = \mu$, за умови, що образ може породити даний фрагмент, і 0 – в іншому разі. Крім того, передбачається, що $p_{ix} = \lambda < \mu$ для всіх i, а кожен відстежуваний фрагмент на зображенні може породжуватися як окремим образом, так і шумом. Усього на зображенні присутньо n_i фрагментів. За таких припущень для обчислення функції правдоподібності необхідно лише знати, які фрагменти породжені образом, а які – шумом. До слова, візьмемо функцію правдоподібності зображення при даному способі та припустимо, що n_p фрагментів породжені зазначеними образами, а n_p-n_i фрагментів – шумом, Запишемо:

$$P(\text{інтерпретація / образ}) = \lambda^{n_p} \mu^{(n_i - n_p)}, \qquad (2.8)$$

зауважимо, що дана величина є більшою для великих значень n_p. З іншого боку, зважаючи на те, що будь-який фрагмент не кожен образ може породити, максимально доступний вибір n_p залежить від обраного способу. Прийнятий метод вказівки є рівнозначним до вибору способу з максимально можливою правдоподібністю за умови даної (простої) породжувальної моделі.

Звідси виникає джерело певних труднощів: якщо образ мало правдоподібний, слід посилатися на додаткову (апріорну) інформацію. До того ж, деякі фрагменти внаслідок шуму можуть породжуватися легше за інші. Якщо цей факт не врахувати, то під час підрахунку рішень деякі образи знаходитимуться в більш вигідному становищі. І наостанок, при даному об'єкті деякі фрагменти можуть бути більш ймовірними за інші. Наприклад, значно частіше кути зустрічатимуться на образі шахової дошки, ніж на зображенні смуг зебри.

2.4.5 Доопрацювання породжувальної моделі

Врахувавши все наведене вище і певним чином удосконаливши найпростішу модель, де фрагменти з'являються незалежно за умови наявності образу, уявімо, що є N різних типів фрагментів. Форсайт і Понс [6] зазначають, що фрагменти кожного типу зі своєю ймовірністю генеруються усіма зображеннями і шумом. Також припустимо, що на зображенні присутні n_{il} зразків фрагментів l-го типу, серед яких n_k генерується образом, а решта – шумом. Зважаючи на те, що при даному способі фрагменти виникають незалежно, а шум від способу не залежить, то правдоподібність дорівнюватиме

$$P(\text{фрагменти породжені образом} \mid j\text{-й образ}) \\ P(\text{фрагментів породжено шумом}). \qquad (2.9)$$

Перший член дорівнює

$$P(\text{тип 1} \mid j\text{-й образ})^{n_1} P(\text{тип 2} \mid j\text{-й образ})^{n_2} \ldots \qquad (2.10)$$

$$P(\text{тип } N \mid j\text{-й образ})^{n_N} P(\text{шум}), \qquad (2.11)$$

котрий можна оцінити як

$$P_{1j}^{n_1} P_{2j}^{n_2} \ldots P_{Nj}^{n_N}. \qquad (2.12)$$

Нехай фрагменти породжуються шумом незалежно один від одного. За такої умови запишемо шумову складову як:

$$P(\text{тип1} \mid \text{шум})^{(n_{i1}-n_1)} \ldots P(\text{тип1} \mid \text{шум})^{(n_{iN}-n_N)}, \qquad (2.13)$$

що можна оцінити у вигляді:

$$P_{1x}^{(n_{i1}-n_1)}...P_{Nx}^{(n_{iN}-n_N)}. \quad (2.14)$$

Відповідно правдоподібність можна представити таким чином:

$$P_{1j}^{n_1} P_{2j}^{n_2} ... P_{Nj}^{n_N} P_{1x}^{(n_{i1}-n_1)}...P_{Nx}^{(n_{iN}-n_N)}. \quad (2.15)$$

Для кожного типу фрагментів k розглядаються два випадки: якщо $p_{kj} > p_{kx}$, то максимум досягається при $n_k = n_{ik}$; у протилежному разі максимум досягається при $n_k = 0$. Подамо через π_i апріорну ймовірність того, що містить j-й образ, а через π_0 – апріорну ймовірність того, що не містить жодного об'єкта. Таким чином, для кожного типу образів j максимальне значення апостеріорної ймовірності представимо як:

$$\left(\prod_m p_{mj}^{n_{im}} \prod_l p_{lx}^{n_{il}} \right) \pi_j, \quad (2.16)$$

де m – індекс, який використовують для підрахунку значень показників, у котрих $p_{mj} > p_{mx}$;

l – для підрахунку значень показників, у котрих $p_{lj} < p_{lx}$.

Така величина формується для кожного об'єкта, а потім з них обирається та, що набула найбільшого апостеріорного значення. Зауважимо, що дана модель – реляційна, хоча, насправді, геометричні ознаки, що зв'язують образи, ми не обчислюємо. Це аргументується тим, що фрагменти зв'язуються умовними ймовірностями їхньої появи у межах цього образу, до того ж, відрізняються для різних образів.

2.4.6 Зазначення зв'язків

З метою покращення простої стратегії зазначення доцільно застосувати геометричні зв'язки. Фрагмент може співставлятися з об'єктом у разі, якщо з об'єктом співставлені сусідні фрагменти і всі вони утворюють відповідну конфігурацію. Термін "відповідна конфігурація" сприйматися як джерело труднощів, однак умовно припустимо, що об'єкти співставлені з точністю до плоского обертання, трансляції і масштабу (таке припущення діє наприклад, для виду особи анфас). Далі припустимо, що даний фрагмент відповідає деякому об'єкту. Отже, беремо *p* найближчих фрагментів і перевіряємо, чи відповідає більш 50 % з них тим же об'єктам, а також чи співпадають кути між трійками узгоджених фрагментів з відповідними кутами об'єкта. Все це можна назвати напівлокальними умовами. Якщо ці дві перевірки пройдено, то реєструється покажчик на об'єкт, поданий цим фрагментом. Зауважимо, що ймовірнісну інтерпретацію даного підходу досить важко побудувати, оскільки ймовірності появи фрагментів тепер залежать від того, де в образі вони розміщені, а також від того, в який спосіб вони згенеровані.

2.4.7 Зазначення на тривимірні об’єкти

Зважаючи на те, що підхід Шміда і Мора було описано з точки зору двовимірних образів, його відносно легко спрямувати на розпізнавання тривимірних виробів. З огляду на це, кожен набір проекцій об'єкта Форсайт і Понс [6] розглядають як інший двовимірний образ. За наявності достатньої кількості проекцій такий підхід є дієвим, оскільки невеликі зміни в двовимірному зображенні, спричинені незначними змінами кута відстеження, компенсуватимуться дозволеним рівнем помилок процесу узгодження інваріантних локальних потоків і кутів.

Згадана стратегія відомостей тривимірного розпізнавання досить активно застосовується до двовимірного, втім має і свої труднощі. Основною проблемою можна назвати велику кількість моделей, що може призвести до

ускладнення процедури зазначення. До того ж, невідомо, яку мінімальну кількість проекцій необхідно для узгодження в подібній схемі.

2.5 Штучні нейронні мережі та їхнє використання під час ідентифікації зображень

Штучними нейронними мережами (ШНМ) називають математичні моделі, а також їхні програмні чи апаратні реалізації, створені за принципом організації і функціонування біологічних нейронних мереж – мереж нервових клітин живого організму. Дане поняття з’явилося під час вивчення процесів, які відбуваються в мозку, а також спроби змоделювати ці процеси.

ШНМ – це система з'єднаних і взаємодіючих між собою простих процесорів (штучних нейронів). Такі процесори, як правило, доволі прості, особливо в порівнянні з процесорами, якими обладнані персональні комп'ютери. Кожен процесор подібної мережі реагує лише на сигнали, що він періодично отримує, а також сигнали, котрі він періодично надсилає іншим процесорам. Незважаючи на те, що будучи об’єднані в достатньо велику мережу з керованою взаємодією, такі локально прості процесори здатні виконувати разом доволі складні завдання. Просту штучну нейронну мережу продемонстровано на рисунку 2.2.

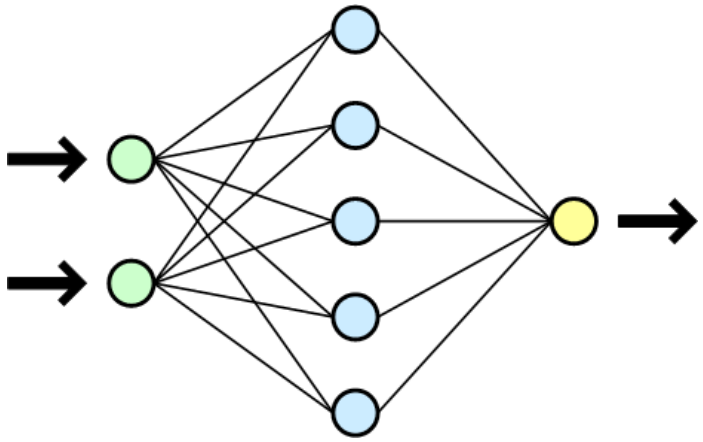

Рисунок 2.2 – Проста нейронна мережа

З точки зору машинного навчання, нейронною мережею вважається окремий випадок методів розпізнавання образів, дискримінантного аналізу, методів кластеризації тощо. З математичної точки зору, навчання нейронних мереж – це багатопараметрична задача нелінійної оптимізації. З точки зору кібернетики, нейронна мережа актуальна для задач адаптивного керування, а у галузі робототехніки як алгоритми. З точки зору розвитку обчислювальної техніки та програмування, нейронна мережа застосовується як метод розв'язання проблеми ефективного паралелізму. З точки зору штучного інтелекту, ШНМ є основою для філософської течії конективізму та ключовим напрямком у структурному підході щодо вивчення можливості створення (моделювання) природного інтелекту із залученням комп'ютерних алгоритмів.

Нейронні мережі, в прямому сенсі цього слова, не програмуються, а навчаються. Така можливість (навчання) є однією з головних переваг нейронних мереж над традиційними алгоритмами. Технічно навчання здійснюється шляхом знаходження коефіцієнтів зв'язків між нейронами. Під час навчання нейронна мережа спроможна виявляти складні залежності між вхідними та вихідними даними, а також проводити узагальнення. Таким чином, за умови успішного навчання мережа зможе повернути правильний результат спираючись на дані, що були відсутні в навчальній вибірці, а також використовуючи дані, що вважаються неповними і/або «зашумленими», частково викривленими.

Образами, у даному випадку, можуть обиратися різні за своєю природою об'єкти: символи тексту, зображення, зразки звуків тощо. Під час навчання мережі надаються різні приклади образів із зазначенням класу їхньої приналежності. Як правило, подають зразок на кшталт вектора значень ознак. Підкреслимо, що сукупність усіх рис повинна однозначно визначати клас, до якого належить зразок. У разі, якщо кількість ознак виявиться недостатньою, мережа почне співвідносити один і той же зразок із декількома класами, відповідно, мова йде про хибний варіант. Після

закінчення навчання мережі можна надати раніше невідомі образи, а у відповідь отримати інформацію про їхню приналежність до певного класу.

Топологія такої мережі пояснюється тим, що кількість нейронів у вихідному шарі, здебільшого, дорівнює кількості визначених класів. До того ж, встановлюється відповідність між виходом нейронної мережі та його класом. Коли до мережі надходить деякий образ, на одному з її виходів має з'явитися показник щодо приналежності образу цьому класу. Водночас, на інших виходах повинна бути показник про відсутність образу у даному класі. Якщо на двох чи більше виходах з'являється показник приналежності до класу, то таке сповіщення інформує про «невпевненість» мережі своїй відповіді [11].

2.6 Розпізнавання об'єктів і їхня інтерпретація

Процес розмітки, в якому алгоритми розпізнавання ідентифікують кожен об'єкт сцени і привласнюють йому мітки (гайковий ключ, перемичка) називається розпізнаванням. У більшості промислових систем машинного зору, здебільшого, передбачається, що об'єкти площини сегментовані у вигляді окремих елементів. Інше загальне обмеження спрямоване на розташування пристроїв збору інформації щодо досліджуваної площини (розміщуються вони, як правило, перпендикулярно робочій поверхні). Можна говорити про зменшення відхилень у характеристиках форми. До того ж активізується процес спрощення сегментації й опису в результаті зменшення ймовірності загороджування одних об'єктів іншими. Керування відхиленнями в орієнтації об'єкта залежить від вибору дескрипторів, інваріантних до обертання, або реалізується завдяки використанню головних осей об'єкта для орієнтування його у напрямку, сформульованому раніше.

Сучасні методи розпізнавання можна розподілити на такі основні категорії, що відповідають теоретичним та структурним методам. У теоретичних за основу береться кількісний опис (статична структура),

водночас структурні методи базуються на символічних описах та їхніх зв'язках (послідовності напрямків у межі, закодованої засобами ланцюгового коду).

Процес, який дозволяє системі МЗ набути більш глибоких знань про навколишнє середовище в порівнянні зі знаннями, що були отримані за допомогою методів, які ми опрацювали у попередніх підрозділах, називається інтерпретацією. Вже розглянута з цієї точки зору інтерпретація характеризує дані методи як невід'ємну частину процесу розуміння зорової площини. Втім для в області МЗ вона подається як власне об'єкт активних досліджень, хоча й значних досягнень поки не отримано. Далі ми стисло опрацюємо проблеми, окреслені сучасними дослідженнями в цій області машинного зору.

Потужність МЗ ґрунтується на здатності виділяти з площини корисну інформацію за різних умов відстеження, а також залученні мінімальних знань про об'єкти площини. Через низку завад (зокрема, геометрії відстеження, нерівномірне висвітлення, наявності тіл, котрі загороджують об'єкти) такий тип обробки порівнюється з важкою задачею. Значна увага приділяється методам зменшення розкиду в інтенсивності. Скориставшись способами зворотного та структурованого освітлення, можна усунути труднощі, що виникають у зв’язку з довільним освітленням робочого простору. За приклад таких складнощів візьмемо тіньові афекти. У свою чергу вони ускладнюють процес визначення контурів і неоднорідності на гладких поверхнях. Це призводить до того, що вони розпізнаються як окремі об'єкти. Очевидно, що більшість окреслених проблем викликано тим, що доводиться працювати з відносно незначною кількістю інформації про моделювання властивостей освітлення і відображення тривимірних площин. Методи розмітки ліній, а також з'єднань вважаються незначними зусиллями в цьому напрямку, втім вони не в змозі кількісно пояснити ефекти взаємодії освітлення і відображення. Значно перспективний підхід започатковано на математичних

моделях, які описують пріоритетні зв'язки між освітленням, відображенням і властивостями поверхні, наприклад, як орієнтація.

2.7 Теоретико-множинна модель розпізнавання та ідентифікації

Обговоримо теоретико-множинну модель процесу розпізнавання та ідентифікації виробів поточного виробництва.

Припустимо, задано простір S, в якому фігурує множина виробів $X = \{x_1 \dots x_n\}$, також в ньому перебуває робот R, обладнаний системою машинного зору S_R. Нехай кожному об'єкту x_i відповідає набір ознак $\{f_{1i} \dots f_{ki}\}$.

У такому разі множину виробів $X = \{x_1, x_2 \dots x_n\}$ можна сформулювати матрицею ознак:

$$\mathrm{F} = \begin{bmatrix} f_{11} & f_{12} & \dots & f_{1n} \\ f_{21} & f_{22} & \dots & f_{2n} \\ \dots & \dots & \dots & \dots \\ f_{n1} & f_{n2} & \dots & f_{nn} \end{bmatrix}.$$

Процес ідентифікації складатиметься з того, що об'єкт x_i може бути розпізнаний за допомогою підмножини ознак f_i' із множини ознак F. Для всієї множини виробів X набір підмножин f_i' реалізує множину F':

$$\mathrm{F}' = \begin{bmatrix} f'_{11} & f'_{12} & \dots & f'_{1k} \\ f'_{21} & f'_{22} & \dots & f'_{2k} \\ \dots & \dots & \dots & \dots \\ f'_{n1} & f'_{n2} & \dots & f'_{nk} \end{bmatrix}.$$

Загальний вигляд процедури розпізнавання виробів формально можемо сформулювати як:

$$R(S) \times \mathrm{X} \Rightarrow \mathrm{X}',$$

де $R(S)$ – робот, обладнаний системою датчиків S;

X – множина виробів;

X′ – множину розпізнаних виробів.

Процедуру розпізнавання та ідентифікації виробів можемо представити як:

$$\mathrm{R(S)} \times \mathrm{X} \equiv \begin{bmatrix} \mathrm{S}_1 \\ \mathrm{S}_2 \\ \dots \\ \mathrm{S}_n \end{bmatrix} \times \begin{bmatrix} f_{11} & f_{12} & \dots & f_{1m} \\ f_{21} & f_{22} & \dots & f_{2m} \\ \dots & \dots & \dots & \dots \\ f_{m1} & f_{m2} & \dots & f_{mm} \end{bmatrix} \Rightarrow \begin{bmatrix} f'_{11} & f'_{12} & \dots & f'_{1k} \\ f'_{21} & f'_{22} & \dots & f'_{2k} \\ \dots & \dots & \dots & \dots \\ f'_{m1} & f'_{m2} & \dots & f'_{mk} \end{bmatrix} \Rightarrow \mathrm{X}',$$

де $\begin{bmatrix} \mathrm{S}_1 \\ \mathrm{S}_2 \\ \dots \\ \mathrm{S}_n \end{bmatrix}$ – множина сенсорів;

$\begin{bmatrix} f_{11} & f_{12} & \dots & f_{1m} \\ f_{21} & f_{22} & \dots & f_{2m} \\ \dots & \dots & \dots & \dots \\ f_{m1} & f_{m2} & \dots & f_{mm} \end{bmatrix}$ – множина ознак;

$\begin{bmatrix} f_{11} & f_{12} & \dots & f_{1m} \\ f_{21} & f_{22} & \dots & f_{2m} \\ \dots & \dots & \dots & \dots \\ f_{m1} & f_{m2} & \dots & f_{mm} \end{bmatrix}$ – множина розпізнаних ознак;

X′ – множина розпізнаних виробів.

У результаті процедура ідентифікації базуватиметься на визначенні в матриці розпізнаних тих ознак, яких вистачатиме для ідентифікації виробів:

$$I \times \begin{bmatrix} f'_{11} & f'_{12} & \dots & f'_{1k} \\ f'_{21} & f'_{22} & \dots & f'_{2k} \\ \dots & \dots & \dots & \dots \\ f'_{m1} & f'_{m2} & \dots & f'_{mk} \end{bmatrix} \Rightarrow \begin{bmatrix} f''_{11} & f''_{12} & \dots & f''_{1l} \\ f''_{21} & f''_{22} & \dots & f''_{2l} \\ \dots & \dots & \dots & \dots \\ f''_{m1} & f''_{m2} & \dots & f''_{ml} \end{bmatrix} = X'',$$

де $\begin{bmatrix} f''_{11} & f''_{12} & \dots & f''_{1l} \\ f''_{21} & f''_{22} & \dots & f''_{2l} \\ \dots & \dots & \dots & \dots \\ f''_{m1} & f''_{m2} & \dots & f''_{ml} \end{bmatrix}$ – матриця ідентифікованих ознак, яка описує множину ідентифікованих виробів X''.

Проаналізовану модель можна застосовувати під час розробки підсистем розпізнавання, а також ідентифікації як складової систем машинного зору. Її реалізація дозволить здійснити розв'язання завдань щодо розпізнавання й ідентифікації об'єктів у робочій зоні маніпуляційних і мобільних роботів.

Основними методами є:

− метод порівняння з еталоном (встановлення збігу двох точкових зображень);

− методи теорії графів і розпізнавання (подання сегментів контуру у вигляді графа і пошуку на графі шляху найменшого значення, що відповідає значущим контурам);

− кореляційний метод (обчислення взаємокореляціоної функції між еталоном і зображенням);

− розпізнавання шляхом зв'язку шаблонів (узгодження компонентів зображення як шаблон і визначення, які об'єкти присутні, проаналізувавши запропоновані зв'язки між знайденими шаблонами);

−штучні нейронні мережі (навчання мережі за різними зразками образів із зазначенням того, до якого класу вони належать).

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ МЕТОДІВ ОБРОБКИ ІНФОРМАЦІЇ В СИСТЕМАХ МАШИННОГО ЗОРУ

3.1 Ключові підходи до практичної реалізації методів обробки інформації

3.1.1 Методи просторової області

Розглянемо два ключові підходи до попередньої обробки інформації. Перший з них засновано на методах просторової області, а другий – на методах частотної області, ґрунтуючись на перетворенні Фур'є. У поєднанні ці підходи охоплюють більшість з сучасних алгоритмів попередньої обробки інформації, що втілені в системах МЗ роботів [12].

До просторової області можна зарахувати сукупність пікселів, які утворюють зображення. Методи просторової області формулюються процедурами, що оперують безпосередньо з цими пікселями. Функції попередньої обробки просторової області подаються як:

$$g(x, y) = h[f(x, y)], \tag{3.1}$$

де $f(x, y)$ – вхідне зображення;

$g(x_y y)$ – вихідне (оброблене) зображення;

h – оператор функції, окреслений у деякій області (x, y). Оператор h можна використати також до низки вхідних зображень для формування, наприклад, суми пікселів K зображень при зменшенні шуму.

Головним підходом під час визначення країв точки (x, y) є використання квадратної або прямокутної області частини зображення з центром у точці (x, y). Центр цієї частини зображення рухається від пікселя до пікселя, розпочинаючи, наприклад, з лівого верхнього кута, водночас для отримання g $(x_y y)$ оператор застосовується для кожного положення (x, y).

Утім іноді залучають й інші форми країв (на кшталт кола), пріоритетними є квадратні форми завдяки легкості їхньої реалізації.

Один із найчастіше затребуваних методів просторової області засновано на використанні так званих масок згортки (чи шаблонів, вікон або фільтрів). Здебільшого, маскою вважають невелику двовимірну систему із розмірністю, наприклад, 3×3, коефіцієнти якої вибирають за принципом, щоб виявити задану властивість зображення.

Уявімо, що дано зображення з постійною інтенсивністю, котре створено з окремо віддалених один від одного пікселів із відмінною від фону інтенсивністю. Протягом дії такого процесу центр маски рухається певним чином по зображенню. Під час виявлення збігу центру маски з положенням кожного пікселя відбувається множення значень усіх пікселів, які знаходяться під маскою, на відповідний коефіцієнт на масці. Згодом підсумовуються результати виконаних дев'яти множень. Якщо всі пікселі під маскою набувають однакових значень (постійний фон), то сума відповідатиме нулю. Якщо ж центр маски знаходитиметься над точкою з іншою інтенсивністю, сума буде відмінною від нуля. У разі перебування зазначеної точки поза центром сума також буде відмінною від нуля, втім на меншу величину. Цю різницю можна усунути шляхом порівняння значення суми з граничним значенням.

Звертаємо увагу, що використання країв не обмежується областями розмірністю 3×3 і випадками, котрі проаналізуємо в подальшому, а саме: зниження шуму, отримання змінних порогів зображення, підрахунок вимірювань параметрів зображення та формування структури об'єкта.

3.1.2 Методи частотної області

Однією зі складових частотної області є сукупність комплексних пікселів у вигляді перетворення Фур'є від зображення. Поняття «частота» доцільно залучати під час інтерпретації перетворення Фур'є , воно виходить з того факту, що результат даного перетворення є сумою синусоїд. З огляду на

підвищенням вимог щодо обробки результатів методи частотної області зрідка використовуються в машинному зорі роботів, як методи просторової області [12].

Утім перетворення Фур'є набуває великого значення під час аналізу руху об'єкта, а також описі об'єкта. До того ж, значна кількість просторових методів щодо поліпшення якості чи відновлення зображення ґрунтується на концепціях перетворення Фур'є.

Дозволяється проводити власне процедуру обчислення двовимірного перетворення Фур'є, послуговуючись тільки одновимірними БПФ-алгоритми. Перетворення Фур'є можна залучати для розв’язання багатьох завдань систем машинного зору. Наприклад, уявивши межі об'єкта у вигляді одновимірного масиву точок, обчисливши їх перетворенням Фур'є, отримуємо значення, що можна використати як опис форми межі. До того ж, одномірне перетворення Фур'є є ефективним під час виявлення руху об'єкта.

Використання дискретного двовимірного перетворення Фур'є припустиме за умов зміни, збільшення або відновлення зображень, втім, як вже було наголошено, реалізація такого методу в промислових системах МЗ досі обмежена через складність необхідних обчислень. У підсумку відзначимо, що двовимірне аналогове перетворення Фур'є може застосовуватись (зі швидкістю світла) оптичними засобами. Такий підхід, який вимагає потребує прецизійного оптичного устаткування, реалізується в промислових умовах для розв’язання певних завдань, зокрема, перевірки якості поверхні металу після фінішної обробки.

3.2 Бібліотека Integrated Performance Primitives (IPP)

Intel IPP розкриває свої можливості у таких функціях, як:

- кодування чи декодування відео;
- кодування чи декодування аудіо;
- JPEG/JPEG2000;

- машинний зір;
- криптографія;
- стиснення даних;
- перетворення кольору;
- обробка зображення;
- трасування променя / візуалізація;
- обробка сигналів;
- кодування мови;
- розпізнавання мови;
- обробка рядків;
- векторна / матрична математика.

Бібліотека використовує розширені набори інструкцій процесора MMX, SSE, SSE2, SSE3, SSSE3, SSE4 разом із багатоядерними.

Intel IPP розподілено між трьома основними групами: сигналами (лінійні масиви даних або вектори), зображеннями (двомірні масиви для типових колірних просторів) і матрицями (nxm масиви для матричних операцій).

Половина функцій корисна для матричних операцій, третина – для обробки сигналів і залишок – для зображень. Функції Intel IPP розбито на 4 типи даних: 8u (8-бітові беззнакові), 8s (8-бітові зі знаком), 16s, 32f (32-бітові з плаваючою точкою), 64f тощо. Здебільшого, розробники застосунків воліють працювати тільки з одним домінуючим типом даних для більшості функцій обробки. Тільки наприкінці обробки перетворення переходять у вихідний формат [12].

3.3 Бібліотека AviCap

Застосунки, розроблені для запису звукових і відеоданих, можуть використовувати зручні високорівневі інтерфейси, що надаються бібліотекою avicap.dll. розробляючи застосунок для запису відео, необхідність піклування

про внутрішню структуру avi-файлів, компресії (стисненню) даних при записі, інтерфейс із драйверами пристрою (або пристроїв) записів виключається. У разі необхідності можна скористатися інтерфейсом нижчого рівня, наявному в бібліотеці avifile.dll.

Для розробки застосунків з метою запису відео, найкраще вибрати клас вікна AVI Cap з бібліотеки avicap.dll. Створивши вікно, попрацювавши з класом AVI Cap, застосунок отримає з подальшим користуванням простий інтерфейс для запису відео та звукових даних до avi-файлу, для попереднього перегляду відео та виконання інших операцій.

У класі AVI Cap передбачені засоби динамічного перемикання пристроїв запису відео та звуку. Його зручність доводиться у тих випадках, коли можливо почергово використовувати декілька таких пристроїв, установлених на комп'ютері. Застосунок може створити avi-файл, призначений для запису, скопіювати вміст одного avi-файлу до іншого, визначити частоту кадрів, вивести на екран діалогову панель, в якій користувач матиме змогу задати формат запису. Крім того, є засоби для роботи з палітрами й універсальним буфером обміну Clipboard.

Для запису звуку клас вікна AVI Cap послуговується засобами бібліотеки mmsystem.dll.

3.4 Бібліотека OpenCV

3.4.1 Оператори для виявлення зображень за допомогою бібліотеки OpenCV

Open CV (англ. Open Source Computer Vision Library, бібліотека комп'ютерного зору з відкритим вихідним кодом), іншими словами, це бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом. Створена на базі C / C ++, до того ж, розробляється для Python, Ruby, Matlab та інших мов.

Open CV призначена для підвищення обчислювальної ефективності процедур обробки відеозображення з пріоритетом на застосування в задачах реального часу.

Open CV написана на мові С, оптимізована належним чином, може користуватися перевагами багатоядерних процесорів. Для роботи з розширеними можливостями бібліотеки рекомендується встановити Intel Performance Primitives (IPP), що дозволить підвищити продуктивність процедур бібліотеки.

Вона дозволяє досить швидко й ефективно розв’язувати складні алгоритми МЗ. Бібліотека наділена більше 500 функціями для реалізації програм у багатьох областях:

- контроль якості продукції, що випускається;
- обробка зображень у медицині;
- забезпечення безпеки;
- інтерфейс користувача;
- робототехніка.

До складу OpenCV належить бібліотека загальних функцій штучного інтелекту «Machine Learning Library» (MLL). Вона спрямована переважно для розпізнавання фрагментів зображення та кластеризації.

Функції бібліотеки використовують для:

- затвердження загального стандартного інтерфейсу комп'ютерного зору для застосунків у цій області;
- сприяння збільшенню кількості таких застосунків і створення нових моделей використання PC;
- привабливості платформ Intel для розробників таких застосунків завдяки додатковому прискоренню Open CV за допомогою Intel® Performance Libraries (Зараз містять IPP (засоби низького рівня бібліотеки для обробки сигналів, зображень, а також медіакодеки) та MKL (спеціальна версія LAPACK та FFTPack));

− автоматичного виявлення Open CV присутність IPP і MKL та їхнього застосування для прискорення обробки.

Ключові модулі.

Ядро cxcore здійснює:

− базові операції над багатовимірними числовими масивами;

− матричну алгебру, математичні функції, генератори випадкових чисел DFT, DCT;

− запис / відновлення структури даних у / з XML/YAML;

− базові функції 2D-графіки;

− підтримку значно складних структур даних: розріджені масиви, динамічно зростаючі послідовності, графи.

CV окреслюється як модуль обробки зображень і комп'ютерного зору. До його функцій належать:

− базові операції над зображеннями (фільтрація, геометричні перетворення, перетворення колірних просторів тощо);

− аналіз зображень (вибір відмінних ознак, морфологія, пошук контурів, гістограми);

− структурний аналіз (опис форм, плоскі розбиття);

− аналіз руху, спостереження за об'єктами;

− виявлення об'єктів чи осіб;

− калібрування камер, елементів відновлення просторової структури.

Highgui визначається як модуль для введення / виведення зображень та відео, створення призначеного для користувача інтерфейсу.

Його доцільність простежується під час виконання таких операцій:

− отримання відео з камер та відеофайлів, зчитування / запис статичних зображень;

− функції для організації простого UI (наразі всі демо-застосунки використовують HighGUI).

Cvaux поєднує у собі як експериментальні, так і застарілі функції.

Даний модуль реалізує:

- просторовий зір: стереокалібрацію, власне калібрацію;
- пошук стереовідповідності, кліки в графах;
- пошук та опис рис обличчя;
- порівняння форм, побудова скелетонів;
- приховані Марківські ланцюги;
- опис текстур.

Оператор Собеля обчислює перші, другі, треті чи змішані похідні зображення за допомогою розширеного оператора.

Функція обчислює похідну зображення, скручуючи зображення з відповідним ядром.

$$dst(x,) = d^{xorder+yorder} src / dx^{xorder} dy^{yorder} |_{(x,y)} . \quad (3.2)$$

Оператори Собеля комбінують Гаусове згладжування з диференціюванням, після чого результат стає більш-менш стійким до спотворення. Як правило, функція викликана, щоб обчислити спочатку *x*- або *y*- похідну зображення. Перший варіант (*X*-оператор) відповідає

$$\begin{vmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{vmatrix},$$

другий варіант (*Y* -оператор)

$$\begin{vmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{vmatrix},$$

або

$$\begin{vmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{vmatrix},$$

в залежності від початку координат зображення. У зв’язку з тим, масштабування не виконувалось, зображення адресата здебільшого має більший розмір абсолютної величини, ніж вихідне зображення.

Ядра взаємодіють з кожним пікселем зображення: піксель розміщують у центрі ядра, водночас значення інтенсивності в сусідніх точках множаться на відповідні коефіцієнти ядра. Наприкінці процедури отримані значення підсумовуються. X-оператор Собеля, що був задіяний до 3×3 матриці вихідного зображення, дає величину горизонтальної складової градієнта інтенсивності в центральній точці цієї матриці, а *Y*-оператор Собеля визначає величину вертикальної складової градієнта [12]. Коефіцієнти ядра обираються за таким принципом: під час його застосування одночасно мають виконуватися в одному напрямку згладжування, а обчислення просторової похідної – в іншому. Величина градієнта підраховується як квадратний корінь з суми квадратів значень горизонтальної та вертикальної складових градієнта.

В результаті утворюється масив чисел. Він відповідає за зміни яскравості в різних точках зображення. Потім активізується операція порівняння з порогом і визначається положення елементів зображення з найбільш сильними перепадами яскравості. Вибір порога є одним з пріоритетних питань виділення перепадів [12].

За результатами обробки отримуємо бінарну матрицю, в якій точки зі значним перепадом яскравості відповідають одиницям, а нулям – всі інші. Як додаткову міру в боротьбі з шумом і ліквідації можливих розривів в контурах активізують морфологічні операції.

У бібліотеці комп'ютерного зору OpenCV такий оператор представлено функцією

void cvSobel(const CvArr* src, CvArr* dst, int xorder, int yorder, int aperture_size=3),

де src – вихідне зображення;

dst – отримане зображення;

xorder – похідна за координатою *x*;

yorder – похідна за координатою *y*;

aperture_size – розмір апертури зображення (апаратура в оптиці – чинний отвір оптичного приладу, котрий відповідає розмірам лінз або діафрагм).

Оператор Лапласа. Розрахуємо ймовірність попадання випадкової величини, розподіленої за нормальним законом, до заданого інтервалу:

$$P(a < X < b) = \int_a^b f(x)dx = \frac{1}{\sigma\sqrt{2\pi}} \int_a^b e^{-\frac{(x-m)^2}{2\sigma^2}}, \qquad (3.3)$$

позначимо

$$\frac{x-m}{\sigma\sqrt{2}} = t; \frac{a-m}{\sigma\sqrt{2}} = \alpha; \frac{b-m}{\sigma\sqrt{2}} = \beta, \qquad (3.4)$$

тоді

$$P(a < X < b) = \frac{1}{\sigma\sqrt{2\pi}} \int_a^b e^{-t^2} \sigma\sqrt{2}dt = \frac{1}{\sqrt{\pi}} \int_a^b e^{-t^2} dt = \frac{1}{2}[\Phi(\beta) - \Phi(\alpha)]. \qquad (3.5)$$

Оскільки інтеграл $\int_a^b e^{-t^2} dt$ не виражається через елементарні функції, то додатково введемо функцію

$$\Phi(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt, \qquad (3.6)$$

яка визначається як функція Лапласа або інтеграл ймовірностей.

Значення цієї функції за наявності різних значень x пораховані та наведені в спеціальних таблицях.

На рисунку 3.1 продемонстровано графік функції Лапласа.

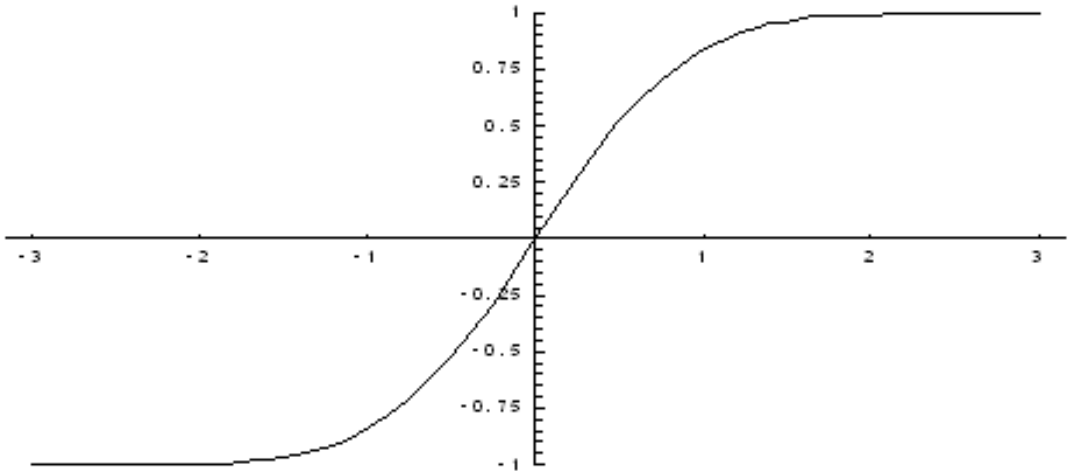

Рисунок 3.1 – Графік функції Лапласа

Функція Лапласа наділена такими властивостями:

$$\Phi(0) = 0,$$

$$\Phi(-x) = -\Phi(x),$$

$$\Phi(¥) = 1.$$

Звертаємо увагу, що функцію Лапласа щн називають функцією помилок і позначають erf x.

До того ж, застосовується нормована функція Лапласа, що пов'язана з функцією Лапласа, нижче на рисунку 3.2 наведено графік нормованої функції Лапласа.

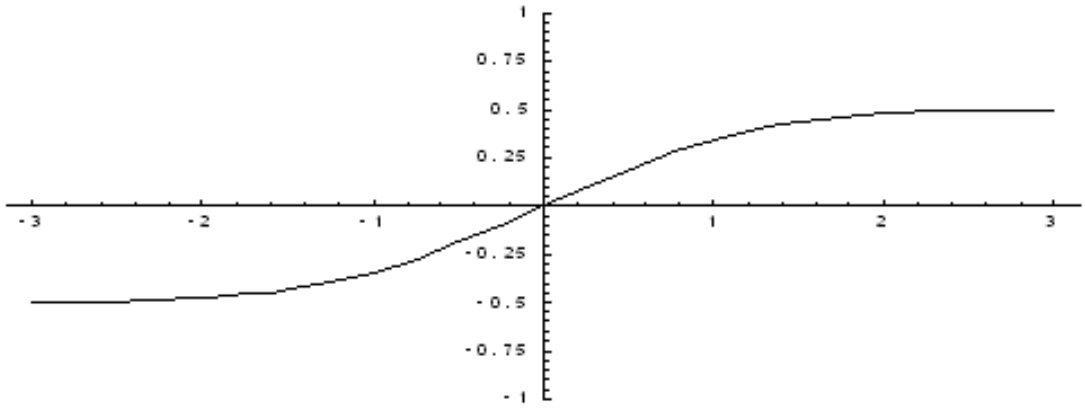

Рисунок 3.2 – графік нормованої функції Лапласа

У бібліотеці комп'ютерного зору OpenCV такий оператор представлено через функцію

void cvLaplace(const CvArr* src, CvArr* dst, int aperture_size=3),

де src – вихідне зображення;

dst – отримане зображення;

aperture_size – розмір апертури зображення (аналогічний до функції cvSobel). Відмінність функції Лапласа від функції Собеля полягає в тому, що в функції Лапласа розташована друга похідна, а в функції Собеля – перша.

Алгоритм Кенні. Кути характеризують контури об'єктів на зображенні, тому сприймаються як вагома проблема під час обробки зображень.

Кути на зображенні трактуються як області з різкими стрибками інтенсивності кольору за умови переходу від одного пікселя до іншого. Виділити кути на зображенні доцільно для значного скорочення обсягу даних під час обробки, при цьому слід зберегти ключові структурні властивості вихідного зображення.

Алгоритм Кенні виділення кутів на зображенні багато хто знає як оптимальний детектор кутових точок. Мета Кенні – поліпшення детекторів,

які були створені значно раніше, ніж почалась розробка. Науковець успішно реалізував поставлені перед собою завдання, виклавши свої ідеї та методи.

Цей алгоритм є актуальним і досі, бо є клюсовим для багатьох сучасних алгоритмів виділення кутів.

У своїй роботі Кенні послуговувався деяким списком критеріїв для поліпшення сучасних методів виділення кутів.

Першим і найбільш очевидним критерієм для нього був низький рівень похибки. Дуже важливо, щоб дотримувалась точність кутових точок зображення і, відповідно, щоб не виникало посилань на НЕ кутові точки. Другим критерієм назвемо хороший рівень локалізації кутових точок: відстань між знайденими кутовими точками та дійсними кутами повинна зводитись до мінімуму. Третім критерієм уважатимемо вимогу, щоб кожному куту відповідало не більше однієї точки. Останній критерій виділили, оскільки перших двох було недостатньо для повного виключення можливості множинного співвідношення різних точок одному і тому ж куті.

Зважаючи на всі ці критерії, детектор кутів Кенні на початку згладжує зображення для зменшення шуму. Після чого знаходить градієнт зображення в кожній точці для виділення областей з найбільшою величиною просторової похідної. Потім алгоритм проходить цими областями та глушить пікселі з не максимальним значенням градієнта (процедура nonmaximum suppression). Масив градієнтів зменшується процедурою гістерезису (лат. hysteresis), що застосовується для обробки залишку пікселів, які не були приглушені. Процедурі гістерезису притаманні два порога. Якщо величина градієнта менша за перший поріг, то піксель встановлюється в нуль (не кут). Якщо величина градієнта більша за другий поріг, то піксель встановлюється в максимальне значення (робиться кутом). Якщо величина розташована між двома порогами, то піксель не встановлюється в нуль, поки не знайдено шлях від даного пікселя до пікселя з величиною градієнта більший за другий поріг.

У бібліотеці комп'ютерного зору OpenCV даний оператор представлено функцією:

void cvCanny(const CvArr* image, CvArr* edges, double threshold1,

double threshold2, int aperture_size=3),

де image – вхідне зображення;

edges – зображення для виділення країв із залученням функції;

threshold1 – перший поріг зображення;

threshold2 – другий поріг зображення;

aperture_size – розмір апертури зображення.

Детектор Харріса. За останні два десятиліття було створено безліч різних детекторів точкових особливостей зображень. Найпопулярнішими з них стали детектор Харріса та детектор за мінімальним власним значенням. У даній роботі залучено обидва.

Опрацюємо детектор за мінімальним власним значенням на прикладі першого зображення послідовності. Для кожного пікселя (x_0, y_0) розраховуємо наступну матрицю:

$$M = \sum_{x,y \in S} w(x,y) \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) \\ \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) & \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix} \tag{3.7}$$

де $I(x,y)$ – яскравість у точці (x,y);

$w(x,y)$ – вагова функція по краю $S(x_0, y_0)$, якою, як правило, є розподіл Гаусса.

У точки, край якої схожий на кут, матриця M матиме два великих позитивних власних значення. Таким чином, умову схожості точки на кут можна записати як:

$$F = \min(\lambda_1, \lambda_2) > 1. \tag{3.8}$$

Далі залишається знайти локальні максимуми функції опису кута $F(x, y)$, що відповідатимуть особливим точкам.

Щоб спростити підрахунки, Харріс запропонував відмовитися від обчислення власних значень матриці, і замість них ввів таку функцію опису кута:

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det(M) - k(Trace(M))^2 . \qquad (3.9)$$

У бібліотеці комп'ютерного зору OpenCV такий оператор представлено функцією

void cvCornerHarris(const CvArr* image, CvArr* harris_responce,

int block_size, int aperture_size=3, double *k*=0.04),

де image – вхідне зображення;

harris_responce – зображення для виведення функції, що повинно мати розмір ідентичний з вхідним зображенням;

block_size – розмір блоків;

aperture_size – розмір апертури зображення;

k – вільний параметр детектора Харріса, його величина, як правило, знаходиться в межах $(0,04 \div 0,15)$ і визначається емпірично.

Далі виконуються аналогічні дії: пошук локальних максимумів, і введення порогового значення.

На другому етапі (етап відстеження (Tracking) опрацьовуються зміщення для кожної особливої точки між сусідніми кадрами, до того ж, обчислюється середній за всіма точками вектор зсуву для кожної пари кадрів.

Основну ідею завдання відстеження визначимо як пошук країв у поточному кадрі, максимально схожого на край точки в попередньому кадрі. Інший важливий момент базується на тому, що даний пошук краю виконується не по всьому зображенню, а в деякому радіусі від початкового положення особливої точки. Тут доцільно застосувати припущення, що за час

між двома кадрами в площині великих змін не відбувається, тому немає сенсу шукати точку далеко від поточного її положення.

Таким чином, якщо в попередньому кадрі координати особливої точки позначити як (x_0, y_0), то в області пошуку R слід шукати вектор $(\Delta x, \Delta y)$, відповідно сума по краю $S(x_0, y_0)$ буде мінімальною:

$$\min_{\Delta x, \Delta y < R} \left[\sum_{x,y \in S(x_0,y_0)} \left(I_n(x,y) - I_{n-1}(x+\Delta x, y+\Delta y) \right)^2 \right]. \qquad (3.10)$$

де $S(x_0, y_0)$ – край порівняння;

R – область пошуку особливості,

I_n і I_{n-1} – яскравість точки в поточному і попередньому кадрах.

Функцією пошуку контурів стає функція cvFindContours (таблиця 3.1):

int cvFindContours(CvArr* image, CvMemStorage* storage, CvSeq** first_contour, int header_size=sizeof(CvContour), int mode=CV_RETR_LIST, int method=CV_CHAIN_APPROX_SIMPLE, CvPoint offset=cvPoint(0,0)) ,

де image – вихідне 8-бітове одноканальное зображення. Ненульові пікселі фіксуються як 1, нульові залишаються 0 – розглядається як бінарний файл. Для отримання такого бінарного зображення з відтінками сірого, можна активізувати функції cvThreshold, cvAdaptiveThreshold або cvCanny. Функція змінює вміст вихідного зображення;

Storage – комірка пам'яті, що містить отримані контури;

first_contour – вихідний параметр, який містить вказівник на перший знайдений контур;

header_size – розмір послідовності заголовка >=sizeof(CvChain), якщо метод CV_CHAIN_CODE, і >=sizeof(CvContour) в іншому випадку;

Mode – мода.

Таблиця 3.1 – Параметри функції виділення контурів

CV_RETR_EXTERNAL	якщо необхідні тільки крайні зовнішні контури
CV_RETR_LIST	виділення всіх контурів і переміщення їх до списку
CV_RETR_CCOMP	виділення всіх контурів і переміщення їх в подвійну ієрархію
CV_RETR_TREE	отримати всі контури та реконструювати повну ієрархію вкладених контурів

Method – метод апроксимації (для всіх модів, окрім CV_RETR_RUNS, який застосовує вбудовану апроксимацію);

I offset позначає, що кожен контур точки зрушився. Його активізують, якщо контури витягуються із зображень, після чого їх слід проаналізувати в контексті цілого зображення.

Методи апроксимації для функції визначення контурів наведені у таблиці 3.2.

Таблиця 3.2 – Методи апроксимації для функції визначення контурів

CV_CHAIN_APPROX_NONE	перетворює всі крапки з ланцюжка на точки
CV_CHAIN_APPROX_SIMPLE	тискає горизонтальних, вертикальних і діагональних сегментах, тобто функція залишається тільки припинення їх центри
CV_CHAIN_CODE	вихідні контуру в ланцюжковому коді Фрімена
CV_CHAIN_APPROX_TC89_L1, CV_CHAIN_APPROX_TC89_KCOS	застосування одного з апроксимаційних ланцюжкових кодів Тех-Чину
CV_LINK_RUNS	використовувати інший алгоритм пошуку контуру за допомогою горизонтальних зв'язків сегментів в один. Тільки CV_RETR_LIST може бути використаний з цим методом

Функція наближення полігональної кривої (кривих) із заданою точністю.

CvSeq* cvApproxPoly(const void* src_seq, int header_size, CvMemStorage* storage, int method, double parameter, int parameter2=0);

src_seq – послідовність ряду точок; header_size – розмір заголовка кривої; storage- контейнер, який апроксимується контурами. Якщо NULL, то застосовується осередок з вхідними послідовностями; method – метод апроксимації; тільки CV_POLY_APPROX_DP, який описує метод Дугласа-Пеукера; parameter – параметр спеціального методу; CV_POLY_APPROX_DP для точної апроксимації; parameter2 – якщо src_seq – одинична послідовність, то параметр повинен наближатися для всіх послідовностей на тому ж рівні або нижче src_seq. Якщо src_seq – масив (CvMat *) точок, то параметр виявляє, чи буде крива замкнутою.

Функції зменшення та збільшення розміру зображень:

Void cvPyrUp(const CvArr* src, CvArr* dst, int filter=CV_GAUSSIAN_5x5);

де src – вихідне зображення;

dst – входить зображення (обов’язково довжина та ширина);

filter – тип фільтра; тільки CV_GAUSSIAN_5x5 підтримується.

Порогову функцію в бібліотеці комп'ютерного зору Open CV визначимо як:

void cvThreshold(const CvArr* src, CvArr* dst, double threshold,
double max_value, int threshold_type);

де src – вхідний масив (8-бітний, 32-бітний);

dst – вихідний масив (однакового типу з вхідним);

threshold – порогове значення;

max_value – максимальне значення, що застосовується з CV_THRESH_BINARY і CV_THRESH_BINARY_INV;

threshold_type – тип порогової функції.

Пошук кіл у відтінках сірого зображення за допомогою перетворення Хафа;

CvSeq* cvHoughCircles(CvArr* image, void* circle_storage, int method, double dp, double min_dist, double param1=100, double param2=100, int min_radius=0, int max_radius=0);

Image – вхідне 8-бітове зображення; circle_storage – комірка, що зберігає виявлені кола. Це може бути сховищем (у даному випадку послідовність кіл створюється за умови зберігання та повертається функцією), або один рядок / один стовпець матриці (CvMat *) типу CV_32FC3, до якого пишуться параметри кіл. Якщо circle_storage – матриця, а фактична кількість ліній перевищує її розмір, максимально можлива кількість кіл повертається. Кожне коло кодується 3 числами з плаваючою точкою: центр координатами (*x*, *y*) і радіусу; Method – реалізовано тільки метод CV_HOUGH_GRADIENT; Dp – як встановити акумулятор, який застосовуються для виявлення центрів кіл. Наприклад, якщо він дорівнює 1, акумулятор матиме той же дозвіл, як у вихідного зображення, якщо це 2 – акумулятор буде в два рази меншим за шириною та висотою тощо; min_dist – мінімальна відстань між центрами окреслених кіл. Якщо параметр буде занадто малим, то кілька сусідніх кіл можуть бути виявлені помилково на додаток до чинних. Якщо він буде занадто великим, то деякі кола можуть бути втрачені; param1 – перший метод конкретних параметрів. У разі CV_HOUGH_GRADIENT це вищий поріг, край детектора (нижче 1 буде в два рази менше); param2 – другий метод конкретних параметрів. У разі CV_HOUGH_GRADIENT – акумулятор. Чим параметр менше, тим більше помилкових кіл можна виявити; min_radius – мінімальний радіус шуканих кіл; max_radius – максимальний радіус.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИЗНАЧЕННЯ ПАРАМЕТРІВ ВИРОБІВ ЗА ДОПОМОГОЮ МАШИННОГО ЗОРУ

4.1 Основні особливості розробленого програмного забезпечення

Метою розробки даного програмного забезпечення є дослідження програмних методів розпізнавання та ідентифікації простих виробів за допомогою бібліотеки комп'ютерного зору OpenCV.

З точки зору MFC проект буде складатися з двох класів: CApp (клас додатка) і CMainWin головне вікно) з відповідною першому функцією InitInstance(); конструктором і картою повідомлень для класу вікна.

У програмі вказується глобальне оголошення функції обробки кадру – void mycallback (IplImage *img).

Вказівниками на зображення будуть: IplImage *image1,*src2,*dst,*dst2,*dst3,*dst4,*gray,*dst5,*dst6,*dst7,*dst8.

Далі виділяються змінні для збереження пам'яті. В даному випадку це CvMemStorage *storage,*storage2;

Конструктор головного вікна програми необхідно описати в такий спосіб:

```
CMainWin::CMainWin()
{ Create (NULL, "OpenCV");
```

Далі в конструкторі вказується дескриптор головного вікна програми:

```
HWND w= this->GetSafeHwnd();
```

Також в конструкторі визначається кількість камер за допомогою команди int ncams=cvcamGetCamerasCount();

Далі йде перевірка наявності камер, якщо їх кількість нульове, виконується ряд функцій, пов'язаних з ініціалізацією камери, основних вікон програми і властивостей зображень:

```
if (ncams) {        bCreate=true;
VidFormat vidFmt={800,600,20.0};
cvcamSetProperty(0,CVCAM_PROP_ENABLE,CVCAMTRUE);
cvcamSetProperty(0,CVCAM_PROP_CALLBACK,mycallback);
cvcamSetProperty(0,CVCAM_PROP_WINDOW,&w);
cvcamSetProperty(0,CVCAM_PROP_SETFORMAT,&vidFmt);
cvNamedWindow(cvGetWindowName(w),CV_WINDOW_AUTOSIZE);
}
```

Для організації виведення результатів обробки візуальної інформації використовується функція cvNamedWindow("Canny", 1), де 1 – ідентифікатор вікна. Результат кожного перетворення вихідного зображення виводиться за допомогою команди cvShowImage().

За допомогою функції cvCreateTrackbar створюється смуга прокрутки cvCreateTrackbar("CannyTrack","Canny",&cannyt,200,NULL) і задається розмір вікна для функції cvResizeWindow("Canny",320,200).

Аналогічним чином створюється вікно для виведення функції визначення контурів.

Потім проводиться ініціалізація камери, або виводиться повідомлення про помилку:

```
if(!cvcamInit())    MessageBox("Error");
```

в іншому випадку проводиться запуск камери:

```
else    cvcamStart();}
```

або видається повідомлення про те, що вона не знайдена.

Закриття головного вікна програми має забезпечувати закриття всіх програмних вікон (в іншому випадку вони можуть залишитися в пам'яті). Для цього необхідно використовувати функцію void CMainWin::OnClose(),в якій

проводиться зупинка камери (cvcamStop()), закриття всіх вікон (cvDestroyAllWindows()),вихід з режиму камери (cvcamExit())і закриття головного вікна:

```
if(!bCreate) cvReleaseImage(&dst);
                DestroyWindow(); }.
```

4.2 Реалізація функції обробки зображень

Обробка зображень в OpenCV проводиться кадр за кадром, а функції зворотного виклику (mycallback), яка в розробленому програмному забезпеченні виглядає так: void mycallback(IplImage *src)

Зображення, отримане з камери в тексті програми позначено змінною src і виводиться в основне вікно програми.

У додатку використовуються різні типи моделей пам'яті, які відповідають різним типам зображень, наприклад:

```
src2=cvCreateImage(cvSize(src->width,src->height),IPL_DEPTH_8U,3);
dst2=cvCreateImage(cvSize(src->width,src->height),IPL_DEPTH_32F,3);
gray=cvCreateImage(cvSize(src->width,src->height),IPL_DEPTH_8U,1);
```

Для повороту зображення використовувалася функція cvFlip(src, src2).

Для нормальної роботи функції Кенні, необхідно перетворити вихідне зображення в чорно-біле (використовується функція cvCvtColor(src2,gray,CV_RGB2GRAY)), після чого виконати саме перетворення функцією cvCanny(gray,dst3,25,100+cannyt,3).

Далі виконується висновок цього зображення у вікно dst3 за допомогою функції cvShowImage("Canny",dst3).

Однією з найпростіших функцій обробки зображень є порогова функція. У бібліотеці OpenCV вона представлена у вигляді:

```
cvThreshold( tgray, gray, 100, 255, CV_THRESH_BINARY ).
```

Тут tgray – вхідний масив (8- або 32- бітний); gray – вихідний масив такого ж типу як і tgray; 100 та 255 – мінімальне і максимальне значення, яке використовується пороговою функцією; CV_THRESH_BINARY – показує тип порогової функції.

Більш покращеною різновидністю порогової функції є функція cvAdaptiveThreshold (dst3, dst4, 5+thresh2, CV_ADAPTIVE_THRESH_MEAN_C, CV_THRESH_BINARY,3,5).

4.3 Реалізація функцій розпізнавання та ідентифікації

Результатом виконання функції Кенні є зображення, що містить края виробів у вихідному кадрі, і є основою для подальших перетворень, зокрема для виділення контурів.

Розглянемо функцію cvFindContours, яка знаходить контури виробів, що знаходяться у полі зору камери:

cvFindContours(gray, storage, &contours, sizeof(CvContour), CV_RETR_LIST, CV_CHAIN_APPROX_SIMPLE, cvPoint(0,0)).

Тут gray є вихідним зображенням у 8-бітному форматі, у області пам'яті storage зберігаються отримані контури; у масиві contours міститься інформація про першому знайденому контурі; CvContour показує розмір структури даних контуру; параметр CV_RETR_LIST означає, що всі виділені контуру поміщаються у список; параметр CV_CHAIN_APPROX_SIMPLE означає, що функція виділяє центри контурів; cvPoint(0,0) показує, що контури витягуються з зображень і потім повинні бути проаналізовані у контексті цілого зображення.

Для знаходження кількох контурів використовується цикл, поданий нижче:

while(contours)

```
{   result = cvApproxPoly( contours, sizeof( CvContour), storage,
    CV_POLY_APPROX_DP, cvContourPerimeter( contours)*0.02, 0 );
    if( result->total == 4 && fabs( cvContourArea( result,
    CV_WHOLE_SEQ)) > 1000 && fabs( cvContourArea( result,
    CV_WHOLE_SEQ)) <( img->height * img->width/2 ) &&
    cvCheckContourConvexity( result) )
  {  s = 0; for( int i = 0; i < 5; i++ )
    {if( i >= 2) { t = fabs( angle( ( CvPoint*)cvGetSeqElem( result, i ), (
    CvPoint*)cvGetSeqElem( result, i-2 ),
    ( CvPoint*)cvGetSeqElem( result, i-1 ))); s = s > t ? s : t; }       }
    if( s < 0.5 )for( int i = 0; i < 4; i++ )cvSeqPush( squares, (
    CvPoint*)cvGetSeqElem( result, i ));      }
    contours = contours->h_next;    }.
```

Функція cvHoughCircles знаходить окружності на сірому зображенні, використовуючи перетворення Х'ю:

cvHoughCircles(gray, cstorage, CV_HOUGH_GRADIENT, 1, gray->height/16, 8, 10, 4, 50).

Дана функція, яка використовується у програмному забезпеченні параметр gray є 8-бітим зображенням; cstorage – область пам'яті, у якій зберігаються окружності, виявлені функцією; CV_HOUGH_GRADIENT – метод реалізації функції; 1 – стек, який використовується для виявлення центрів кіл з тим же розділенням, що і вихідне зображення; gray->height/16 – обчислюється мінімальна відстань між центрами виявлених окружностей; наступні 2 параметра відповідають за накопичення порогів у виявлених окружностях; останні 2 параметра відповідають за мінімальний і максимальний радіус знаходять окружностей.

Для відображення кіл використовується цикл, який виглядає так:

```
for( int i = 0; i < circles-> total; i++ )
{       float* p = ( float*)cvGetSeqElem( circles, i );
        cvCircle( out, cvPoint( cvRound( p[0]), cvRound( p[1])),
```

```
2, CV_RGB( 200, 0, 0), -1, 8, 0 );
cvCircle( out, cvPoint( cvRound( p[0]), cvRound( p[1])), cvRound(
p[2]), CV_RGB( 200, 0, 0), 1, 8, 0 ); }
```

Відображення прямокутних зображень здійснюється командою void drawSquares(IplImage *img, CvSeq* squares);

Прямокутні об'єкти визначаються за допомогою функції поліліній cvPolyLine(img, &rect, &count, 1, 1, CV_RGB(200,0,0), 1, CV_AA, 0);

Далі читається послідовність 4 ліній

```
CV_READ_SEQ_ELEM( pt[0], reader );
CV_READ_SEQ_ELEM( pt[1], reader );
CV_READ_SEQ_ELEM( pt[2], reader );
CV_READ_SEQ_ELEM( pt[3], reader ).
```

Кут повороту ліній полілінії заданий за допомогою функції:

```
double angle = abs(pt[1].y-pt[2].y)/sqrt((pt[1].x-pt[2].x)*
(pt[1].x-pt[2].x)+(pt[1].y-pt[2].y)*(pt[1].y-pt[2].y)+0.00001);
```

Самі лінії полілінії промальовуються за допомогою функцій:

```
cvLine( img, cvPoint(0,img->height/2), cvPoint(img->width,img->height/2),
CV_RGB(200,200,200),1, 8, 0 );
cvLine( img, cvPoint(img->width/3,0), cvPoint(img->width/3,img->height),
CV_RGB(200,200,200),1, 8, 0 );
cvLine( img, cvPoint(img->width/3*2,0), cvPoint(img->width/3*2,img->height),
CV_RGB(200,200,200),1, 8, 0 );  cvPutText( img, st, pt[1], &font,
CV_RGB(200,0,0).
```

Виведення прямокутних виробів здійснюється функцією drawSquares(out,findSquares4(gray, mainStorage)), де out – зображення, що містить прямокутні фігури.

Отже розроблене програмне забезпечення виконує знімання й обробку інформації, що надходить з WEB-камери (МЗ робота), забезпечує виконання меж, контурів, ідентифікацію простих виробів.

Лістинг програм наведено у Додатках А та Б.

4.4 Промислова безпека та аналіз умов праці на робочому місці

Розміри лабораторії, в якій виконувалась робота, складають 5×6 м. Робоче місце складається з стола, стільця і персонального комп'ютера. У приміщенні працює 4 людини. Площа приміщення 30 м^2, об'єм – 90 м^3. Згідно ДСанПиН 3.3.2.007-98 площа на одне робоче місце має становити не менше 6 м^2, а об'єм – 20 м^3. Для даного приміщення робоча площа і об'єм на одну людину відповідає нормам, так як в нашому випадку площа на одне робоче місце становить 10 м^2, а об'єм – 30 м^3.

Живлення комп'ютерів здійснюється від трифазної чотирьох провідної електричної мережі змінного струму з глухо-заземленою нейтраллю і напругою 220 В, частотою 50 Гц.

Згідно НПАОП 40.1-1.21-98 лабораторію можна віднести до категорії без підвищеної небезпеки, так як в приміщенні відсутні чинники, які викликають підвищену або особливу небезпеку.

Для створення безпечних умов праці необхідно провести ряд організаційних і технічних заходів. Згідно НПАОП 40.1-1.32-01 для запобігання ураження людини електричним струмом в приміщенні застосовується система занулення [15].

Згідно з вимогами НПАОП 0.00-4.12-05 необхідно провести вступний, первинний на робочому місці, повторний, цільовий та позаплановий інструктажі [15]. Зміст інструктажу відповідає вимогам НПАОП 0.00-4.12-05. Інструктаж відзначається в відповідних журналах з підписами інструктованих і інструктора.

Робота в лабораторії проводиться сидячи і не вимагає фізичної напруги. Тому вона відноситься до категорії Іа (легкі фізичні роботи, енерговитрати до 120 ккал / год). З метою забезпечити комфортні умови для

працівників та відповідно до ДСН 3.3.6.042-99 у відвідуванні встановлені наступні метеорологічні параметри [15]:

а) для холодного періоду:

1) температура повітря від 22 °С до 24 °С;

2) вологість повітря від 40 % до 60 %;

3) швидкість руху повітря оптимальна до 0,1 м/с;

б) для теплого періоду року:

1) температура повітря від 23 °С до 25 °С;

2) вологість повітря від 40 % до 60 %;

3) швидкість руху повітря оптимальна до 0,1 м/с.

Для освітлення робочих місць і приміщення в цілому застосовується як природне бічне освітлення, так і штучне освітлення.

Приміщення з ЕОМ повинні мати природне і штучне освітлення відповідно до ДБН В.25-28-2006 «Природне і штучне освітлення». Природне світло повинно проникати через бічні світлові прорізи, зорієнтовані, як правило, на північ або північний схід, і забезпечувати коефіцієнт природної освітленості (КПО) не нижче 1,5 %:

e^{IV}= 1,35, де $e^{IV}_{норм}$ – нормоване значення КПО для 4-го поясу світлового клімату СНД.

Згідно ДСН 3.3.6.037-99 рівень шуму в лабораторії не перевищує 50 дБ.

Загальний рівень штучного освітлення приміщення можна перевірити за допомогою методу питомої потужності.

Розрахункова формула методу [15]:

$$W = \frac{W_{\Sigma}}{S}, \tag{4.1}$$

де W – питома потужність, Вт/м^2;

S – площа приміщення, м^2;

W_{Σ} – загальна потужність освітлювальної установки, Вт, яка розраховується за формулою:

$$W_{\sum} = W_{св} \cdot n_{св}, \tag{4.2}$$

де $W_{св}$ – потужність одного світильника, $W_{св}$ = 80 Вт;

$n_{св}$ – кількість світильників у приміщенні, $n_{св}$ = 4 шт.

Дане приміщення має площу 30 м^2, в якому розташовано шість світильників потужністю 80 Вт:

$$W_{\sum} = 4 \cdot 80 = 320 \text{ Вт},$$

$$W = \frac{320}{30} = 11 \text{ Вт/м}^2.$$

Табличне значення для отриманого результат освітленість складе 200 лк, коли відповідно до стандарту ДБН В.2.5-28-2006. в лабораторії освітленість повинна бути 300-500лк. Для отримання освітленості в 400 лк необхідна питома потужність 21 Вт/м^2.

Для поліпшення умов роботи в лабораторії необхідно в денний час застосовувати додаткове освітлення.

ВИСНОВКИ

Під час написання кваліфікаційної роботи було проаналізовано методи ідентифікації виробів у робототехнічних системах, а також побудовано теоретико-множинну модель розпізнавання й ідентифікації.

Розглянуто ключові методи:

− метод порівняння з еталоном (встановлення збігу двох точкових зображень);

− методи теорії графів та розпізнавання (представлення сегментів контуру у виді графа та пошуку на графі шляху найменшого значення, що відповідає значимим контурам);

− кореляційний метод (обчислення взаємокореляційної функції між еталоном і зображенням);

− разпізнавання через зв’язки шаблонів (узгодження компонентів зображення як шаблон та визначення, які об’єкти є присутніми, опрацювавши запропоновані зв’язки між знайденими шаблонами);

− штучні нейронні мережі (навчання мережі з різних зразків образів із зазначенням того, до якого класу вони належать).

Опрацьовано практичну реалізацію методів обробки інформації у робототехнічних системах. Сформульовано два основні підходи до попередньої обробки інформації. Перший з них базується на методах просторової області, а другий – на методах частотної області з використанням перетворення Фур'є. Разом зазначені підходи охоплюють більшість з чинних алгоритмів попередньої обробки інформації, що застосовуються у системах машинного зору роботів. Головними бібліотеками, що працюють з методами розпізнавання та ідентифікації, названо бібліотеку Integrated Performance Primitives (IPP), бібліотеку AviCap, бібліотеку комп'ютерного зору з відкритим кодом OpenCV. Опрацювали основні функції та їхню реалізацію у бібліотеці OpenCV.

Досліджено та програмно реалізовано методи розпізнавання та ідентифікації простих об'єктів. Пріоритетною бібліотекою розпізнавання обрано бібліотеку комп'ютерного зору OpenCV. Отримані результати реалізують виділення меж та контурів об'єктів, визначення центрів мас. Для простих об'єктів залучені функції розпізнавання та ідентифікації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008:2015 Інформація та документація «Звіти у сфері науки і техніки». Структура та правила офомрлювання. / В. Земцева; Ю. Поліщук, канд. фіз.-мат. наук; Р. Санченко, канд. техн. наук; Л. Шрамко; А. Ямчук (науковий керівник) ДП «УкрНДНЦ» від 22 червня 2015р. № 61 з 2017- 07-01.

2. Методичні вказівки з підготовки кваліфікаційної роботи бакалавра для студентів усіх форм навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми «Автоматизація та комп'ютерно-інтегровані технології» / Упоряд.: І.Ш. Невлюдов, А.О. Андрусевич, О.В. Токарєва, С.П. Новоселов, О.В Сичова. – Харків: ХНУРЕ, 2022. – 55 с.

3. Навчальний посібник з підготовки кваліфікаційної роботи бакалавра для здобувачів вищої освіти денної і заочної форм навчання спеціальностей 151 «Автоматизація та комп'ютерно-інтегровані технології» та 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» освітньої програми «Системна інженерія» : навчальний посібник / І. Ш. Невлюдов, О. В. Токарєва, О. М. Цимбал, А. І. Бронніков ; М-во освіти і науки України, ХНУРЕ. – Харків : Видавництво Іванченка І. С., 2023. – 218 с.

4. Гэри Брадскі. Learning OpenCV. Computer Vision with the OpenCV Library / підручник / Гэри Брадскі, Адріан Кехлер; – O'Reilly Media, 2018. – 580 с.

5. Джосепс Ховс. Android Application Programming with OpenCV / Джосепс Ховс. O'Reilly Media, – 190 с.

6. Невлюдов І.Ш. Основи виробництва електронних апаратів / І.Ш. Невлюдов. – Харків: Компанія СМІТ, 2005. – 592 с.

7. Кравченко Л.С. Розробка технологічного процесу складання: Навч. посібник для студентів машинобудівних спеціальностей. – Харків: НТУ«ХПІ», 2004 р. – 146 с.

8. Невлюдов І. Ш. Виробничі процеси та обладнання об’єктів автоматизації: Підручник для студентів вищих навчальних закладів / І. Ш. Невлюдов. – Кривий Ріг: Криворізький коледж НАУ, 2017р. – 444 с.

9. Невлюдов І.Ш. Технічні засоби автоматизації: Підручник / І.Ш. Невлюдов, А.О. Андрусевич, О.І. Филипенко, Н.П. Демська, С.П. Новоселов. – Кривий Ріг : Криворізький коледж НАУ, 2019. – 366 с.

10. Невлюдов І.Ш. Людино-машинний інтерфейс в технічних засобах автоматизації: Навчальний посібник / І.Ш. Невлюдов, О.І. Филипенко, Б.О. Шостак. – Харків : «ХТМТ», 2019. – 244 с.

11. Невлюдов І.Ш. Основи наукових досліджень / І.Ш. Невлюдов, Ю.М. Олександров, А.О. Андрусевич, О.О. Чала. – Кривий Ріг : КК НАУ, 2017. – 344 с.

12. Лубко Д.В., Шаров С.В. Методи та системи штучного інтелекту: навч. посібник – Мелітополь: ФОП Однорог Т.В., 2019. – 264с

13. Діагностика та контроль робочих процесів: навч. посіб. для студентів спеціальності «Прикладна механіка» денної та дистанційної форм навчання / В. М. Доля – Харків: НТУ «ХПІ», 2019. – 129 с.

14. Rosenblum M. Human Emotion Recognition from Motion Using a Radial Basis Function Network Architecture / M. Rosenblum, Y.Yacoob, L.Davis // IEEE Workshop on Motion of Non-Rigid and Articulated Objects, 2018. – 257 с.

15. Організація керування умовами праці» підготовки освітнього рівня бакалавр усіх спеціальностей та усіх напрямів університету [Електронний ресурс] / ХНУРЕ; розроб.: Т.Є. Стиценко, Г.В. Пронюк, Н.М. Сердюк. – Харків, 2017. – 108 с.